NAVAL POSTGRADUATE SCHOOL

MONTEREY, CALIFORNIA

THESIS

MODELING, SIMULATION AND IMPLEMENTATION OF A NON-COHERENT BINARY-FREQUENCY-SHIFT-KEYING (BFSK) RECEIVER-TRANSMITTER INTO A FIELD PROGRAMMABLE GATE ARRAY (FPGA)

by

Juan P Svenningsen

September 2005

Thesis Advisor: Herschel Loomis
Co-Advisor: Frank Kragh

THIS PAGE INTENTIONALLY LEFT BLANK

REPORT DOCUMENTATION PAGE		Form Approved OMB No. 0704-0188

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	September 2005	Master's Thesis

4. TITLE AND SUBTITLE: Modeling, Simulation and Implementation of a Non-Coherent Binary-Frequency-Shift-Keying (BFSK) Receiver-Transmitter into a Field Programmable Gate Array (FPGA)	5. FUNDING NUMBERS M6890904POH9023
6. AUTHOR(S) Juan P Svenningsen	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey, CA 93943-5000	8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING /MONITORING AGENCY NAME(S) AND ADDRESS(ES) Marine Corps Tactical Systems Support Activity Camp Pendelton, CA	10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

12a. DISTRIBUTION / AVAILABILITY STATEMENT	12b. DISTRIBUTION CODE
Approved for public release; distribution is unlimited.	A

13. ABSTRACT (maximum 200 words)

This thesis presents the use of a field programmable gate array (FPGA) to implement a non-coherent binary-frequency-shift-keyed receiver-transmitter (BFSK-RT) that simulates the modulation of the SINCGARS radio, the RT-1523C. An FPGA successfully, and with very few resources, implemented the desired modulation and demodulation. Topics covered include FPGA history, the hardware and software utilized, a summary of the SINCGARS RT-1523C characteristics, the BFSK-RT on FPGA design procedure and the design results.

14. SUBJECT TERMS SINCGARS, RT-1523C, Field Programmable Gate Array (FPGA), non-coherent receiver design, binary-frequency-shift-keying (BFSK) modulator design, on-off keying (OOK), digital communications	15. NUMBER OF PAGES 108
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	UL

NSN 7540-01-280-5500 Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. 239-18

THIS PAGE INTENTIONALLY LEFT BLANK

Approved for public release; distribution is unlimited.

MODELING, SIMULATION AND IMPLEMENTATION OF A NON-COHERENT BINARY-FREQUENCY-SHIFT-KEYING (BFSK) RECEIVER-TRANSMITTER INTO A FIELD PROGRAMMABLE GATE ARRAY (FPGA)

Juan P Svenningsen
Captain, United States Marine Corps
B.S., United States Naval Academy, 1999

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN ELECTRICAL ENGINEERING

from the

NAVAL POSTGRADUATE SCHOOL
September 2005

Author: Juan P Svenningsen

Approved by: Professor Herschel Loomis
Thesis Advisor

Professor Frank Kragh
Thesis Co-Advisor

Professor Jeffrey B. Knorr
Chairman, Department of Electrical and Computer Engineering

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

This thesis presents the use of a field programmable gate array (FPGA) to implement a non-coherent binary-frequency-shift-keyed receiver-transmitter (BFSK-RT) that simulates the modulation of the SINCGARS radio, the RT-1523C. An FPGA successfully, and with very few resources, implemented the desired modulation and demodulation. Topics covered include FPGA history, the hardware and software utilized, a summary of the SINCGARS RT-1523C characteristics, the BFSK-RT on FPGA design procedure and the design results.

THIS PAGE INTENTIONALLY LEFT BLANK

TABLE OF CONTENTS

LIST OF FIGURES

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF TABLES

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF ABBREVIATIONS AND ACRONYMS

ADC	Analog to Digital Converter
ASIC	Application Specific Integrated Circuit
BFSK	Binary-frequency-shift-keying
BFSK-RT	Binary-frequency-shift-keying Receiver-Transmitter
COMSEC	Communications Security
COTS	Commercial-off-the-Shelf
CPLD	Complex Programmable Logic Device
DAC	Digital to Analog Converter
DSP	Digital Signals Processing
EDA	Electronic Design Automation
EDM	Enhanced Data Mode
EEPROM	Electrically Erasable Programmable Read-only Memory
FIR	Finite Impulse Response
FPGA	Field Programmable Gate Array
FPGA-RT	Field Programmable Gate Array Receiver-Transmitter
GUI	Graphic User Interface
HDL	Hardware Description Language
IC	Integrated Circuit
IEEE	Institute of Electrical and Electronics Engineers
IIR	Infinite Impulse Response
I/O	Input/Output
IOE	Input/Output Element
IP	Intellectual Property
JTRS	Joint Tactical Radio System
LAB	Logic Array Block
LE	Logic Element
LED	Light Emitting Diode
MCTSSA	Marine Corps Tactical Systems Support Activity

MSPS	Mega Samples Per Second
NCO	Numerically Controlled Oscillator
NCOs	Network-centric Operations
OOK	On-Off Keying
PAL	Programmable Array Logic
PCOs	Platform-centric Operations
PLA	Programmable Logic Array
PLD	Programmable Logic Device
RAM	Random Access Memory
RT	Receiver-Transmitter
SDM	SINCGARS Data Mode
SDR	Software Defined Radio
SINCGARS	Single Channel Ground and Airborne Radio System
SOPC	System On a Programmable Chip
SRAM	Static Random Access Memory
VHDL	VHSIC Hardware Description Language
VHSIC	Very High Speed Integrated Circuit

ACKNOWLEDGEMENTS

I’d like to thank my thesis advisors, Professors Herschel Loomis and Frank Kragh, for dedicating time to ensuring I approached this project in a systematic manner. Thanks to Professor Loomis’ intricate knowledge of FPGAs and design techniques, several difficult design problems were quickly solved. Professor Kragh’s communication systems expertise and guidance provided me a starting point from which to start the design and constantly dedicated time to ensuring I understood the system that I developed. Without their guidance, time and effort this work would have not been possible.

I’d also like to thank Capt Rick Paradise for assisting me in the understanding of communication systems and receiver design. He dedicated much of his time to ensure I understood the inner workings of basic RT systems so my design would be in proper working order.

Finally, I’d like to thank Capt Max Greene and the Marine Corps Tactical Systems Support Activity for their support in the form of equipment, expertise, funding and the opportunity to work on this project.

THIS PAGE INTENTIONALLY LEFT BLANK

EXECUTIVE SUMMARY

The goal of this thesis was to implement a receiver-transmitter that simulates the modulation and demodulation of the SINCGARS RT-1523C. The RT was implemented on an Altera® Stratix™ Edition DSP development board with an on-board Stratix FPGA. The designed RT was a non-coherent BFSK-RT.

Descriptions of the FPGA and development board capabilities are presented, to include a brief history of FPGAs. The hardware descriptions are followed by a design flow discussion that describes the possible ways to design systems for implementation on the DSP board used in this thesis. Other background topics include a discussion of the SINCGARS RT-1523C, to include its characteristics and most recent upgrades.

A thorough explanation of how the design was approached is presented. This document describes the procedures taken, including the observed problems and solutions to ensure the design functioned properly. The design software tools used throughout the thesis are MATLAB®'s Simulink®, the Altera DSP Builder and the Quartus II programmer.

Next, the system was implemented onto hardware and tested for software and hardware functionality. The software simulation was conducted in Simulink and the hardware simulation was conducted on the DSP board, using an oscilloscope to observe the transmitted and received waveforms and bit streams.

THIS PAGE INTENTIONALLY LEFT BLANK

I. INTRODUCTION

The single channel ground and airborne radio system (SINCGARS) is currently widely used throughout the United States Marine Corps and Army for terrestrial tactical voice and data communications [1]. The last several years have continued the shift in military operations away from platform-centric operations (PCOs) to network-centric operations (NCOs) [2]. NCOs rely heavily on data communications, and therefore the SINCGARS has been used increasingly for data communications [2]. Unfortunately, the SINCGARS has often not been able to adequately support these NCO data communications demands. The author, while assigned to the U.S. Marine Corps operational forces, has observed that this problem has become much more evident when utilizing the SINCGARS manpack variant. The problem can be traced back to the power and antenna differences between vehicle and manpack variants. Vehicle variants include one or two amplifiers and antennas in excess of 6 feet. The manpack variant has no power amplification and has PRC-77 legacy antennas that are intended for short range transmissions. Therefore, the SINCGARS manpack radio has been found inadequate in providing dismounted troops with satisfactory data traffic capabilities as required by current doctrine.

The use of design software which interfaces with hardware to create a software-defined design has become a more commonplace technique for creating system solutions. One such solution is the use of field programmable gate arrays (FPGAs) which can be programmed using software to perform hardware operations. This solution can allow designers to

examine, change, implement and test systems without working with circuit boards, chips or other electronic parts. [3]

This thesis addresses the use of an FPGA to model, simulate and implement the modulation of the SINCGARS radio, a binary-frequency-shift-keying receiver-transmitter (BFSK-RT). The results of this research can be used to analyze, modify and test BFSK-RT radio designs in order to identify modifications that can improve the performance of the SINCGARS radios, in particular the SINCGARS manpack.

A. SINGLE CHANNEL GROUND AND AIRBORNE RADIO SYSTEM (SINCGARS)

NCOs involve the coordination of diverse combat units to assemble a military capability that is greater than the capability of any one unit. This involves sharing of intelligence, reconnaissance, and surveillance information, collaborative planning, and command and control alignment — all of which require networked communications across the force. Therefore, the greatest demands for military radio capabilities are in effective data transfer as part of the greater network. At this junction in time, the military owns equipment capable of sharing situational-awareness information and administrative reports that keep the commander informed of each subordinate unit’s position and disposition. In order to employ this equipment to its full capability, adequate communications means are required. Currently, the main terrestrial communications system that is employed by the United States military for this task is the SINCGARS. [1]

The SINCGARS has been employed in the United States military since 1987 when it was contracted from ITT Industries. The SINCGARS was purchased to replace the aging

PRC-77 single-channel plain-text radio set, and it was to provide the U.S. Armed Forces with an improved reliable means of voice communications in the battlefield. As needs changed through the years, so did the system. The current SINCGARS, fielded in 1998, is capable of providing single-channel and frequency-hopping communications for voice and data, and it comes equipped with an integrated communications security (COMSEC) device [4]. Test of the SINCGARS show that it is capable of providing communications in excess of 20 kilometers in a stationary position with an adequate antenna [5]. However, the range of the manpack SINCGARS receiver-transmitter unit, the RT-1523C, is reduced to less than one kilometer when transmitting data [6]. This reduced range limits the ability of the radio to effectively support data transfer in the digital battlefield. It is due to this fact that the Marine Corps Tactical Systems Support Activity (MCTSSA) requested this thesis explore the implementation of the radio system in an FPGA, in order to facilitate modifications in the radio hardware design that can improve the performance of the RT-1523C receiver-transmitter radio set. MCTSSA provided an Altera® Stratix™ Professional Edition DSP Development Kit for this task.

B. ALTERA® DIGITAL SIGNALS PROCESSING (DSP) DEVELOPMENT KIT, STRATIX™ PROFESSIONAL EDITION

The increased capabilities of technology and the constant improvement of semiconductors continue to provide programmable logic devices that facilitate the design and implementation process for system solutions that require flexible design methods. These devices allow designers to have flexible hardware solutions, based on programmable logic. The Altera DSP Development Kit, Stratix Edition, is

one such hardware solution for the communications industry. The DSP development kit provides the necessary hardware to develop complete system-on-a-programmable-chip (SOPC) solutions using the Stratix FPGA. [3]

1. Stratix FPGA

The Stratix FPGA is the Altera Corporation solution to requirements for faster and more powerful designs. The SINCGARS RT-1523C has 23 application-specific integrated-circuits (ASICs) [4]. This technology requires redesign and manufacture of an entire ASIC to upgrade that part of the system. In contrast, FPGA-based systems can be updated by re-programming the logic in the FPGA. Through the use of reprogrammable FPGAs, a designer can implement and program the design onto an FPGA, test it and proceed to large-scale production once the testing is concluded satisfactorily. At this point, the FPGA can be added to the radio. Since FPGAs are reprogrammable, the design can be changed or improved at any time after fielding. It is this characteristic that makes FPGAs a good tool to examine the RT-1523C and make changes to it without having to implement and manufacture any ASICs. The use of FPGAs in the RT-1523C will allow the military to use commercial-off-the-shelf (COTS) technology, an advantageous alternative due to the large knowledge base available in the programmable logic industry, and reduced costs of COTS acquisition and life cycle logistical support.

C. SINCGARS HARDWARE RADIO

The RT-1523C SINCGARS radio is a non-coherent binary-frequency-shift-keying (BFSK) hard-decision receiver [5]. The receiver performance has very little recorded analysis in the dismounted variation and because of its ASIC-based design, changes to the radio are difficult and variation in

testing is limited. Using an FPGA-based DSP board, the designer can make changes to the radio architecture based on operating conditions, therefore creating improved performance in different environments.

D. GOALS

The U.S. military has already established a program to provide a replacement for the SINCGARS with software defined radios (SDRs), which comprise the new Joint Tactical Radio System (JTRS). Unfortunately the JTRS manpack contract, cluster five of the JTRS program, was started on July 16, 2004 and is currently in the design and development phase with full production planned to start in late 2009 [7]. This leaves a capability gap in manpack radios because the JTRS is not expected to reach the operating forces until 2010. This assumption does not account for possible program delays and the additional time required for full fielding, typically a few years' time span.

SDRs take advantage of programmable logic technology to make the radios reconfigurable [7]. This same technology can easily be used to improve the RT-1523C while the military awaits the complete fielding of the JTRS manpack.

The purpose of this thesis is to design a non-coherent BFSK receiver, program the FPGA with the design, test the design, and compare the data with the experimental data of the actual SINCGARS manpack. In conjunction with the thesis, "Modeling and Simulation of the Physical Layer of the Single Channel Ground and Airborne Radio System (SINCGARS)," by Captain Richard Paradise, USMC, MCTSSA can use this design to evaluate any possible modifications to the physical layer of the SINCGARS [8].

E. METHOD AND STRUCTURE

The focus of this thesis is implementing a non-coherent BFSK radio using an FPGA, and improving the design to implement the SINCGARS characteristics as much as possible.

The design process was started by first implementing an on-off keying (OOK) transmitter. After the OOK transmitter was designed, a BFSK transmitter was implemented and a non-coherent receiver was created. The radio was designed using the MathWorks Simulink® software, which interfaces with the Altera Quartus® II design software to program the FPGA.

Chapter II gives a brief history of FPGAs, discusses the Stratix FPGA and its characteristics, describes the Altera DSP Development Board and its characteristics, the design software, the design flow and briefly discusses the RT-1523C SINCGARS receiver-transmitter set.

Chapter III describes the steps taken to create the design and program the FPGA. The chapter provides an in-depth look at the use of the software necessary to go from concept to testing of the BFSK system using an FPGA. The tools discussed in this chapter include Simulink, Altera MegaCore® and OpenCore® Plus functions, the MegaCore Compiler, the SignalCompiler box and the DSP Builder.

Chapter IV presents the results of the Simulink and hardware simulations and comparisons of simulated results with the known performance of the SINCGARS.

Chapter V presents a summary of the thesis, conclusions regarding the DSP board and FPGA utility and recommendations regarding the options that may be added to the

SINCGARS design and testing efforts when an FPGA solution is implemented. Emphasis is placed in the suggestions for further research and testing, as well as possible implementation alternatives and possible modifications to the proposed design.

Two appendices are included, Appendix A discusses errors encountered and lessons learned and Appendix B includes all the Simulink models that were created in order to implement the FPGA-RT.

THIS PAGE INTENTIONALLY LEFT BLANK

II. BACKGROUND

This chapter discusses FPGAs, the Altera DSP development kit, design software, design implementation and the SINCGARS RT-1523C.

A. FPGA OVERVIEW

The history of FPGAs began with the implementation of integrated circuits (IC) that could have their logic programmed after the ICs were manufactured. [9]

The first such IC was the programmable logic array (PLA), which had a two-level structure of AND and OR gates with programmable connections. As the programmable device industry grew, PLAs were modified and programmable array logic (PAL) devices were introduced; PALs also have a two-level structure but only the AND array is programmable and the OR array is fixed. PALs and PLAs are called programmable logic devices (PLDs). As IC capabilities increased, programmable logic vendors introduced complex programmable logic devices (CPLD). CPLDs implement multiple PLDs onto a single chip with programmable interconnections, called a switch matrix, thereby increasing the scale of logic possible from the IC. [9]

When CPLDs were being invented, FPGAs were developed with a different approach to achieving a large amount of programmable logic. FPGAs have a large number of individual logic blocks that are smaller than PLDs and a large and programmable connection network that is distributed throughout the chip. This approach allows the designer to maximize the use of the logic elements (LEs) available and does not restrict use of any part of the chip. This is not the case with CPLDs. [9]

In general, the differences between CPLDs and FPGAs are architectural. As previously mentioned, CPLDs are made of PLDs with programmable interconnections called a switch matrix. However, a CPLD switch matrix is not capable of achieving all possible connections, thereby reducing the chances of 100% utilization of all on-board logic [10]. By contrast, FPGA architecture is specifically designed to maximize the use of all on-board logic through the use of the fully programmable interconnections that comprise the majority of the FPGA area. Other major differences include the amount of on-board logic and chip size, both in favor of FPGAs as they are larger and contain more logic elements. Neither is universally more effective than the other, and the fact that the two leading vendors in programmable logic devices, Altera and XILINX®, sell CPLDs and FPGAs show that consumers have applications for both devices. [11] [12]

Altera and XILINX have different approaches to programmable logic products. Altera focuses on design solutions. Altera Corporation not only provides customers with devices and support which includes design software and services, but they also provide development products that allow the designer to implement and test designs [3]. By contrast, XILINX focuses on devices and support. Like Altera, they provide solutions to certain ‘End Markets’, but XILINX does not provide testing platforms [11]. This means XILINX-based solutions are designed and created by the designer, to include the implementation and testing platform. Therefore, given an inexperienced designer or the need for a fast design solution, Altera is a good option. However, if the desired effect is the most effective use of a de-

vice, then XILINX provides a better option because of their robust and varied devices with extensive support [11]. Due to resources provided by the requesting activity, MCTSSA, an Altera FPGA was used.

1. Altera Stratix FPGA

Current FPGA design makes FPGAs an excellent choice for DSP solutions. In 2002, ALTERA introduced to the programmable device market their high-density Stratix FPGA, shown in Figure 1. The Stratix FPGA provides high bandwidth, on-board memory, a high density of logic elements, DSP blocks and high performance input/output (I/O) capabilities. [13]

Figure 1. Stratix FPGA.

The Stratix device achieves these qualities through its “two-dimensional row- and column-based architecture” [14]. The device’s internal logic array blocks (LABs), memory blocks and DSP blocks are arranged in rows and columns. Each LAB has 10 logic elements (LEs). The device has three different types of memory with parity: M512 Random Access Memory (RAM) blocks (576-bit dual-port memory

RAM), M4K RAM blocks (4,608-bit true dual-port RAM), and M-RAM blocks (589,824-bit true dual-port RAM). These memory blocks are located throughout the device logic array, as shown in Figure 2. Also within the device array are two columns of DSP blocks which can implement eight 9 X 9-bit full-precision multipliers that can implement four 18 X 18-bit multipliers or one 36 X 36-bit full-precision multiplier. The outer edges of the device have input/output elements (IOEs) that feed I/O pins. The IOEs contain bidirectional I/O buffers and registers for input, output and output-enable signals. These features allow the device to support numerous I/O standards and provide an interface for external memory devices. Figure 2 shows a subsection of a Stratix FPGA block diagram and how each block is distributed throughout the device. [14]

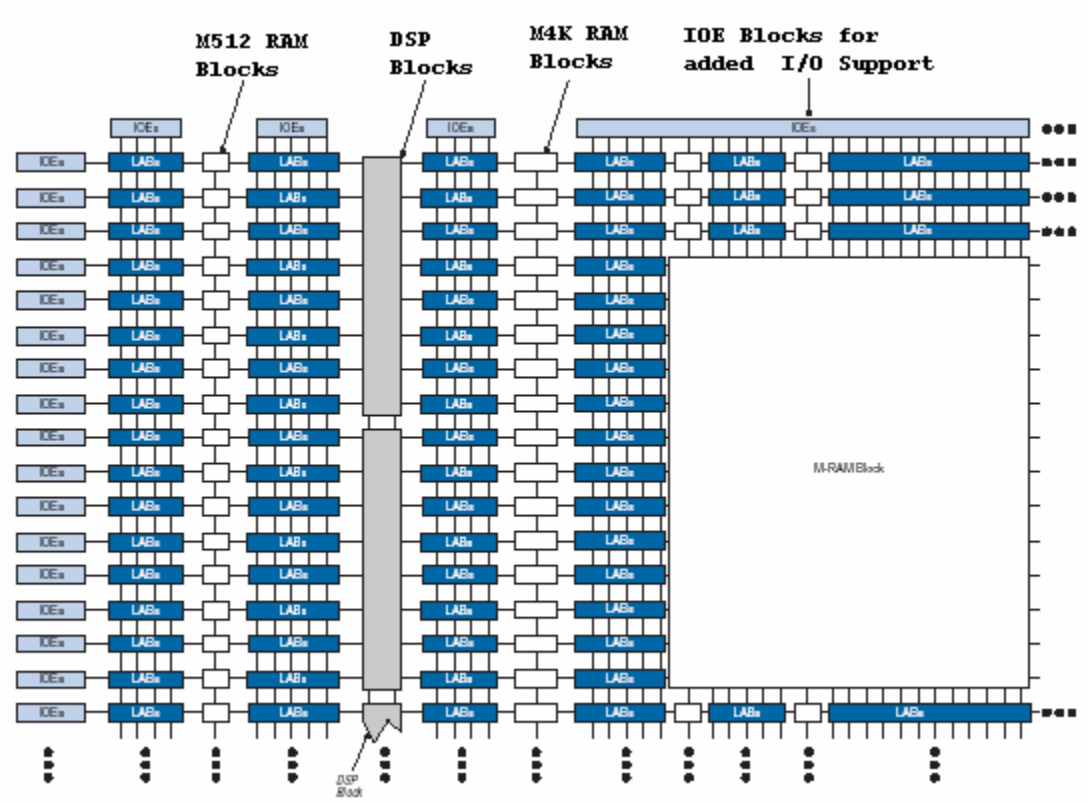

Figure 2. Stratix Device Block Diagram (from [14]).

The device that was used for this thesis was the EP1S80B956-C6, the most powerful of the Stratix devices, which is included in the EP1S80 DSP development board that was provided by MCTSSA. The EP1S80 Stratix FPGA characteristics are summarized in Table 1. [15]

Table 1. EP1S80 Overview (from [15]).

Feature	EP1S80B956-6
Logic elements (LEs)	79,040
M512 RAM Blocks (32 × 18 bits)	767
M4K RAM Blocks (128 × 36 bits)	364
M-RAM Blocks	9
Total RAM bits	7,427,520
DSP Blocks	22
Embedded multipliers (based on 9 × 9)	176
PLLs	12
Maximum user I/O pins	679
Package type	956-pin BGA
Board reference	U1
Voltage	1.5-V internal, 3.3-V I/O

B. ALTERA STRATIX DSP DEVELOPMENT BOARD

The Altera Stratix DSP development board, shown in Figure 3, facilitates design work. The major components include the Stratix FPGA, analog inputs and outputs run by analog-to-digital and digital-to-analog converters, a memory subsystem, a PLD for configuration options, an 80-MHz on-board oscillator and various input and output interfaces. The input components include three pushbuttons, eight dipswitches, and the output components include a dual seven-segment display and two light emitting diodes (LEDs). [15]

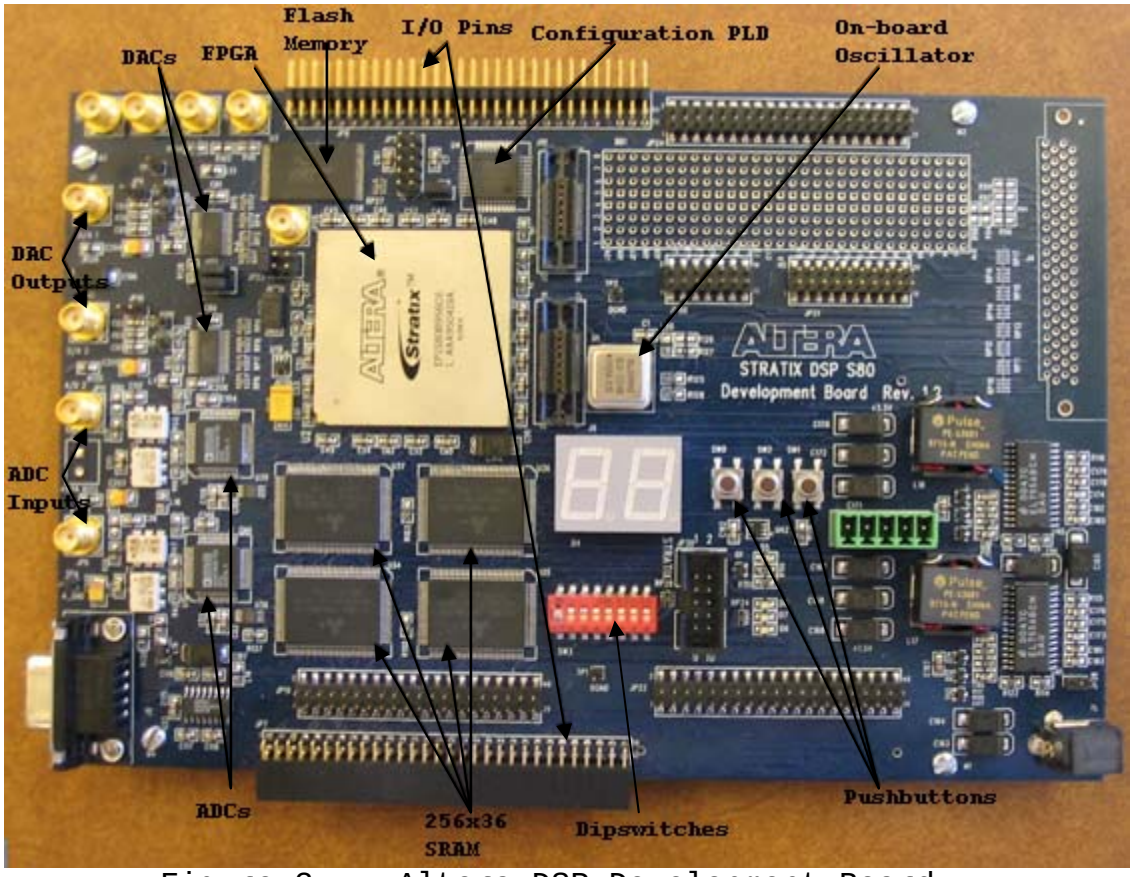

Figure 3. Altera DSP Development Board.

1. Analog I/O

The analog input and output consists of two 12-bit analog-to-digital converters (ADC) and two 14-bit digital-to-analog converters (DAC). The ADCs can sample at a maximum rate of 125 mega-samples-per-second (MSPS) and the DACs convert 1 14-bit number to an analog signal every 6.06 ns. Each ADC input circuit is DC-coupled and its output to the FPGA is in two's-complement format. The D/A converter box in Figure 4. It is best modeled as an ideal current source with output 0-20 mA, proportional to the 14-bit unsigned binary value received from the FPGA. This current is filtered by the two capacitors and resistor to pass a fre

quency range of approximately 16 kHz to 230 MHz. The output signal maximum is 0.5 V when a 50 ohm antenna is attached to the output. [15]

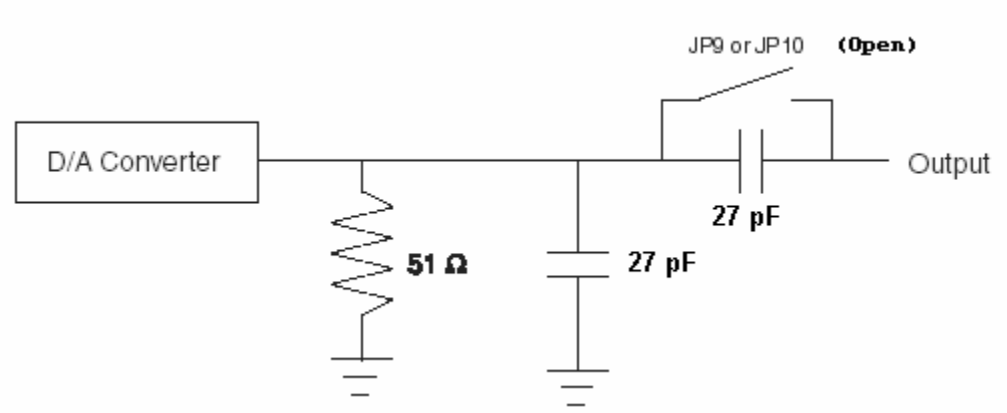

Figure 4. Circuitry after DAC (from [15]).

2. Memory Subsystem

The DSP board embedded memory capabilities consist of 2 megabytes of 7.5 ns synchronous 256 X 36 static random access memory (SRAM) configured as two independent 36-bit wide buses and a single 64-megabit flash memory device [15]. Flash memory, or electrically erasable programmable read-only memory (EEPROM), is a non-volatile memory that does not require power to hold data; this kind of technology can be re-programmed a limited number of times. Therefore it is normally used for data that does not change [9]. By contrast, SRAM is a fast read/write memory that holds the stored values until the memory is re-written or the system is powered down. The arrangement of the SRAM allows the board to support high data rates and concurrent processing by using the 36-bit buses independently. [15]

3. Configuration Options

The flash memory device allows the board to store an on-board configuration when used in combination with the Altera EPM7064 PLD. Since the Stratix FPGA is SRAM-based

device, configuration is required each time the system is powered. The PLD and flash memory combination allows a design to be stored on-board. As soon as the board is turned on, the PLD will configure the FPGA using the data stored in the flash memory. This is called a non-volatile configuration scheme. The PLD can hold two different configurations, a preset factory-configuration and a user-defined configuration. The factory-configuration is a test that allows the user to ensure the board is working properly. Once the user configuration has been programmed on the PLD, the use of jumper JP18 selects the user-defined configuration. [15]

4. I/O Interfaces

The majority of the DSP board is dominated by multiple I/O headers and various connectors that can be used to add expansion devices and analytical devices. The I/O interfaces are: 90 digital evaluation I/O pins, two Mictor-type connectors for logic analyzer debugging, a Texas Instrument board expansion interface, two 40-pin I/O headers for Analog Devices Corporation converters and a prototyping area that allows other devices to be added to the board. All of these interfaces are directly connected to FPGA pins, and can be used to extract signals out of the FPGA for analysis or debugging; specific assignments can be found in the EP1S80 development board data sheet. [15]

C. DESIGN SOFTWARE TOOLS

Given the complexity of PLDs and FPGAs, it is necessary to discuss the software necessary to program these devices with a design. For this purpose, in the mid-1980s, the Institute of Electrical and Electronics Engineers (IEEE) sponsored the development of a hardware description language (HDL) that could document and model a system in a

hierarchical manner. The HDL that was created was the Very High Speed Integrated Circuit (VHSIC) HDL, also known as VHDL. As mentioned earlier, VHDL was initially intended to be a “documentation and modeling language” [9], but after the creation of VHDL synthesis tools by various commercial vendors, VHDL surpassed its initial purpose. These synthesis tools use VHDL files to create a database that the fitter in turn uses to map the circuit design onto the specific FPGA, or other target device. The components created depend on the device that needs to be programmed and most synthesis tools allow the user to provide information regarding the device that will be used. [9]

The tool that makes the transition from software designs to hardware implementations are device fitters. A fitter performs what is known as *place-and-route*, it maps the database from the synthesis onto the device that will hold the design. The device map can then be analyzed for timing, thereby completing the design flow. This is the basic idea behind HDL design flow, and Figure 5 shows a simplified block diagram of a HDL design flow. [9]

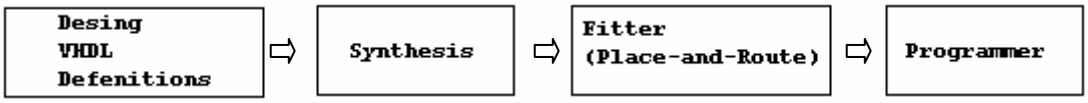

Figure 5. Simplified HDL Design Flow (from [9]).

The Quartus II software is the Altera Corporation tool that takes a design from concept to hardware implementation; it includes tools that range from design entry to programming tools. Altera also has the DSP Builder which links the Quartus II software to the MathWorks MATLAB®/Simulink software to create a DSP Builder design flow. The DSP Builder allows designers to implement a de-

sign in a user friendly graphical environment in Simulink [16]. Figure 6 shows the design flow used in this thesis.

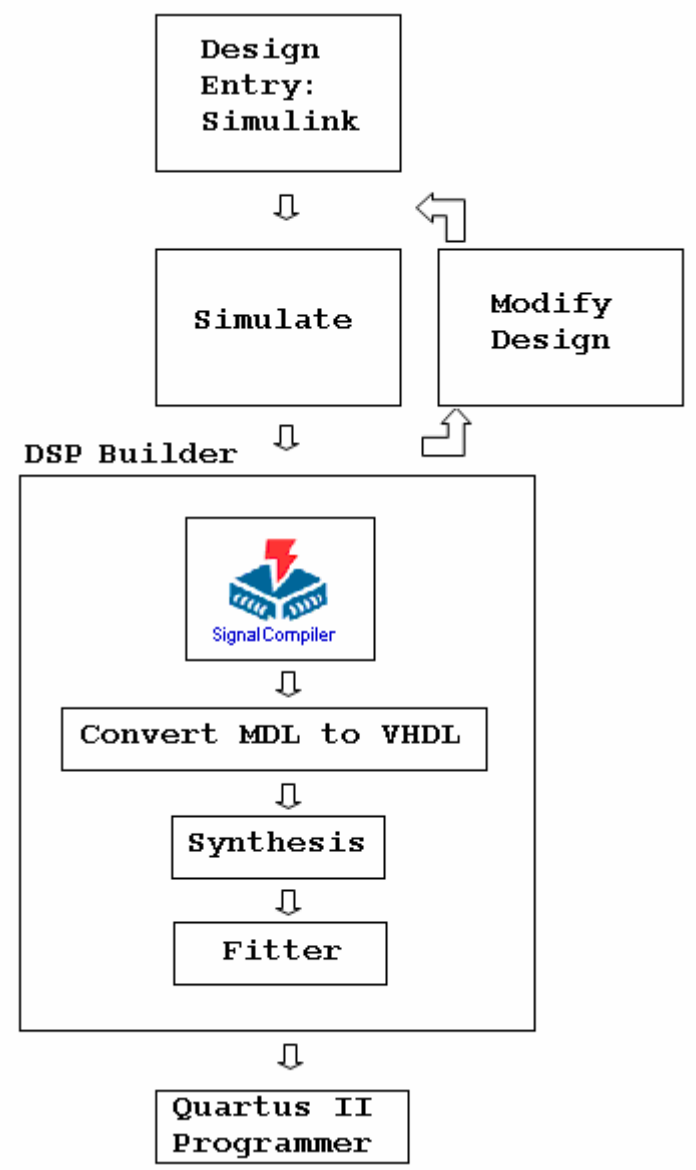

Figure 6. Thesis Design Flow.

1. Design Entry: Simulink

Simulink is a design program that allows users to implement designs graphically. In this thesis, it was used for design entry and simulation. Simulink uses block libraries that contain graphical representations of systems that model controls, signal processing, communications and

other time-varying systems. The DSP blocks from the DSP Builder can be used in the Simulink environment to create a model of the desired design, which can then be used to continue the design flow. [17]

Simulink also allows the creation of hierarchical designs. This is done by creating subsystems within the system model. The use of subsystems simplifies the design because a system solution can then be broken down into subsystem solutions that can be implemented several times into the same model. This feature allows the designer to create more complicated designs that are organized using subsystems within a model. [16]

Simulink's simulation abilities allow the designer to troubleshoot designs and ensure the system functions properly before implementation. It is this property that makes Simulink a good design entry choice for DSP designs. Since Altera blocks are used in Simulink models, the design can be designed, simulated, troubleshot and validated before implementing it in hardware. [17]

2. DSP Builder

The Simulink block libraries contain many predefined configurable blocks that can be used to create a wide variety of system designs. The Altera DSP Builder is a tool that is installed into Simulink, using the MATLAB console, which adds the *Altera DSP Builder* library to the Simulink libraries; the DSP Builder is necessary because it acts as a link between the Simulink and Quartus II software. The DSP Builder library consists of many blocks ranging from arithmetic operations to the Altera MegaCore functions. These blocks must be used in designs that are intended for hardware implementation because the DSP Builder cannot con-

vert Simulink blocks into VHDL, only DSP Builder blocks. The usage of DSP Builder blocks allows the design to be simulated and tested in Simulink to ensure proper system behavior. [17]

The DSP Builder can perform all design implementation tasks from the Simulink environment using the Quartus II software in the background. The DSP Builder does this through the SignalCompiler block which must be used in all DSP Builder models. The steps taken by the SignalCompiler in order to program the DSP board with a model are: 1 – convert model to VHDL, 2 – synthesis, 3 – fit system to Quartus II and 4 – board programming. This design flow allows programming of the device before doing any analysis of the system, which is done in Simulink, not Quartus II. Unfortunately, the use of the MegaCore functions does not allow the SignalCompiler to perform board programming because of MegaCore usage limitations. [17]

The Altera MegaCore functions are called intellectual property (IP) blocks. IPs provide the designer with complex, ready-to-use functions that have been optimized for implementation on Altera devices. Using IPs improves system development time and enhances the performance of the system because IPs are made for use on Altera devices. Since IPs are copyrighted functions, to use IPs a designer must obtain an evaluation license. When evaluation IPs are used, the designer can implement, evaluate, and test the design in software and hardware. However, there are restrictions to IP use if a full license is not purchased. [19]

In order to use IPs with an evaluation license, the hardware has to maintain a connection to the Quartus II

software via a joint actions test group cable. This is called the tethered mode. An IP implementation in hardware will run indefinitely if the hardware and the Quartus II software maintain communications. If the hardware implementation is not linked to the Quartus II software (untethered mode), the IP will only function for a limited time, after which it will be deactivated and the device will have to be programmed again. Therefore, the board programming must be performed from by the Quartus II in tethered mode programmer and not by the SignalCompiler programmer. [19]

a. Convert Model to VHDL

In order to begin the DSP Builder part of the design flow, the SignalCompiler block shown in Figure 6, must be used. Once the SignalCompiler block is double clicked the compiler is started. The compiler automatically checks the accuracy of the model and is ready to begin the model to VHDL conversion. VHDL conversion is necessary because the Quartus II synthesis tools cannot perform synthesis on a model file; the synthesis tool requires VHDL to perform synthesis. [9]

b. Synthesis

The Synthesis part of the design flow is actually performed by the Quartus II software via the DSP Builder, so it is performed from the Simulink environment. Synthesis is the conversion of the VHDL into components that can be assembled in the desired hardware [9]. In the Quartus II software, the synthesis step in the design flow “builds a single project database that integrates all the design files in a design entity or project hierarchy” [18]. The software uses the database throughout the rest of the synthesis and is updated until the database contains the opti-

mized project which is then used for the rest of the design flow. As the database is updated and modified, the software checks for syntax and flow errors. The Quartus II synthesis also ensures the project is efficient by creating assignments of resources on the device being used and minimizing gate count. During synthesis, the project is also evaluated for timing requirements and modified to meet them. [18]

c. Fitter

The Quartus II Fitter performs place-and-route using the database developed during synthesis, matching logic and timing requirements with the resources available onboard the target device. After the logic is assigned to a particular cell, the fitter selects the best path and pin connection assignments to make within the device. The fitter does this using the parameters established by the user, and then optimizes the design. If the design is not feasible an error message is produced and the designer must redesign. [18]

The Quartus II Timing Analyzer runs by default and reports various timing data. This data can then be used to validate the timing parameters for the design [9].

3. Quartus II Programmer

The design flow is completed when the project is compiled and downloaded onto the Altera device. This is done by the Quartus II Programmer. After the Quartus II Fitter step of the DSP Builder design flow is completed, the compiler creates a programming file that the Quartus II Programmer uses to program a device. At this point, the hardware has been programmed with the developed design and is ready for real-time testing in a physical environment. [18]

D. SINCGARS RT-1523C

The SINCGARS RT-1523C, shown in Figure 7, has been in service in the U.S. military since the late 1990s and was upgraded by the RT-1523E, a version that has not been fully fielded yet. The RT-1523C is currently the main communications asset for the U.S. Marine Corps, therefore it was selected as the SINCGARS radio variant to examine. [1]

Figure 7. SINCGARS RT-1523C.

The RT-1523C is a non-coherent binary-frequency-shift-keying receiver-transmitter (BFSK-RT) that is capable of performing data and voice communications [5]. The RT has many important characteristics, shown in Table 2, but the most relevant properties to this thesis are the modulation and bandwidth (channel spacing) [20].

Table 2. RT-1523C Performance Data (from [20]).

PERFORMANCE CHARACTERISTIC	DATA
Operating Voltage	
Manpack	13.5 volts from lithium battery
Vehicular	27.5 volts from vehicular battery
Frequency Range	30 - 87.975 MHz
Number of Operating Frequencies	2,320
Channel Spacing	25 KHz
Frequency Stability	±5 parts per million
Frequency Offset Ability (SC)	±5 and 10 KHz
Type of Modulation	FM
Audio Response Capability	300 - 3,000 Hz
Types of Operation	
Retransmit	Automatic
Remote	Push-to-Talk, Release to Receive
Data	Automatic via Data Device
Modes of Operation	
Voice	SC and FH
Retransmission	SC to SC, SC to FH, FH to FH
Digital data	SC, FH
Remote	With AN/GRA-39, C-11291/VRC, or C-11561(C)/U
Text	Plain or Cipher
Tuning	Electronic. SC frequency entered manually by using keyboard. Up to eight SC channels and six FH channels can be loaded and later selected using CHAN (channel) switch.

ITT implemented improvements to the waveform to improve data-throughput. In order to satisfy backward compatibility, the new waveform is the enhanced data mode (EDM) that is used in combination with the old SINCGARS data mode (SDM). The modification to the waveform made the EDM distinct from the SDM and causes a radio operating in EDM to reject an SDM message and vice versa. As Figure 8. shows, the throughput was drastically improved from 1.5 seconds to 0.432 seconds per data-waveform. [5]

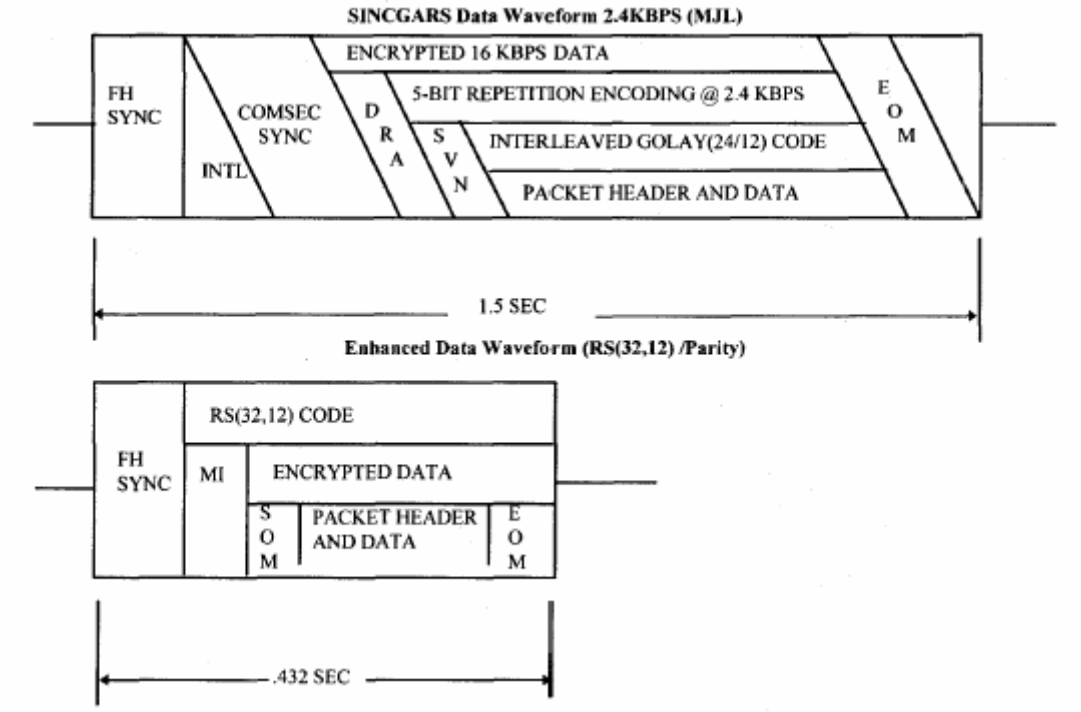

Figure 8. SINCGARS Waveforms (from [5]).

The characteristics of the RT-1523C make it a good tool for voice communications, but weak for data communications. Unfortunately, most if not all of the experimentation that has been performed on the radio has been conducted on units using vehicle mounts or the OE-254 antenna, an impedance matched antenna designed to operate in the 30-88 MHz range in a frequency-hopping mode [5] [6]. Both options provide long ranges and facilitate testing and improve results by maximizing range. The standard manpack radio is the AN/PRC-119C which consists of the RT-1523C, the 10-foot AS-4266/PRC (pole collapsed or extended) antenna and the 3-foot AS-3683/PRC (tape collapsed or extended) antenna [20]. This particular configuration of the SINCGARS has very little published test data regarding data communications.

In August 2004, MCTSSA conducted a field study of the AN/PRC-119C performance using the standard manpack equipment. Communications with a probability of bit error of greater than 10^{-4} was considered unreliable and communications with a probability of bit error of less than 10^{-4} was considered reliable. Both the pole and tape antennas were tested in a collapsed and extended variation with the RT-1523C at medium and high power. Table 3 summarizes the result and recommendations in the report. [6]

Table 3. Manpack Performance Results (from [6]).

Antenna Configuration	Maximum Distance Medium Power (meters)	Maximum Distance High Power (meters)
Pole Collapsed	277	515
Pole Extended	505	1044
Tape Collapsed	223	519
Tape Extended	364	798

From the results of the test we make the following recommendations:

- More experimentation to determine the physical characteristics of the existing antennas. A test using an antenna range to determine gain would help. SPAWAR San Diego has this capability.
- Research improvements in antenna technology and recommend a replacement for existing antennas. Research is already being conducted for wearable antennas, but we do not know if these antennas will perform better or are just more concealable. This has NPS thesis potential.
- Research the propagation methods used in development of System Planning Engineering and Evaluation Device (SPEED) software for coverage analysis. This report could help influence the improvement of this software.
- Investigate improvements to the SINCGARS modulation and coding schemes to improve the BER vs. E_b/N_o curve for data transmission. This research is ongoing at MCTSSA and NPS.

As can be noted from Table 3, the data shows that the AN/PRC-119C manpack needs better range. The most effective configuration was with the 10-foot antenna fully extended transmitting at full power. This option is not acceptable because of the cumbersomeness of conducting foot-mobile operations with a 10-foot antenna on the RT and because operating the RT at full power makes it easier for the enemy to conduct signals intelligence. [1]

After reviewing the results of the MCTSSA tests, it is obvious that the current version of the SINCGARS is incapable of adequately supporting data transfer for foot-mobile units. It is the low range of the AN/PRC-119C manpack that this thesis hopes to assist in solving. With the development of a BFSK-RT FPGA, MCTSSA can further their research efforts and have a way to test possible modulation variations of the RT-1523C.

This chapter presented a brief FPGA and DSP board overview, the design software tools are described and the RT-1523C is presented. This information will serve as a basis for understanding the design flow discussed in the following chapter.

THIS PAGE INTENTIONALLY LEFT BLANK

III. DESIGN FLOW

This chapter describes the design steps taken to implement a BFSK-RT into an FPGA.

A. APPROACH

In order to properly model the SINCGARS, some research was done to identify the actual type of receiver the RT-1523C implements. The RT-1523C receiver was determined to be a non-coherent BFSK receiver after reading Hamilton's "SINCGARS System Improvement Program (SIP) specific radio improvements" article. [4]

Further research on the RT was performed to determine the bit-rate (R_b) and operating frequencies (f_1, f_0). The *U.S. Marine Corps Technical Manual, Intermediate and Depot Maintenance, SINCGARS* states that the RT-1523C operates at an intermediate frequency of 12.5 MHz, and transmits data at 16,000 bits per second (bps). [20] Assuming the RT-1523C performs orthogonal signaling, then the operating frequencies are $f_0 = 12.492$ MHz and $f_1 = 12.508$ MHz. The throughput is less than this in EDR mode due to the need to transmit parity symbols, but the bit rate (including data and parity bits) is always 16 kbps [20].

To simplify understanding and visualization of a successful implementation, this design was constructed with sufficient bandwidth to identify the waveforms on an oscilloscope, but note should be taken of the discrepancy in bandwidth availability in the RT-1523C. Figure 9 shows a basic block diagram of the digital communication system that was designed.

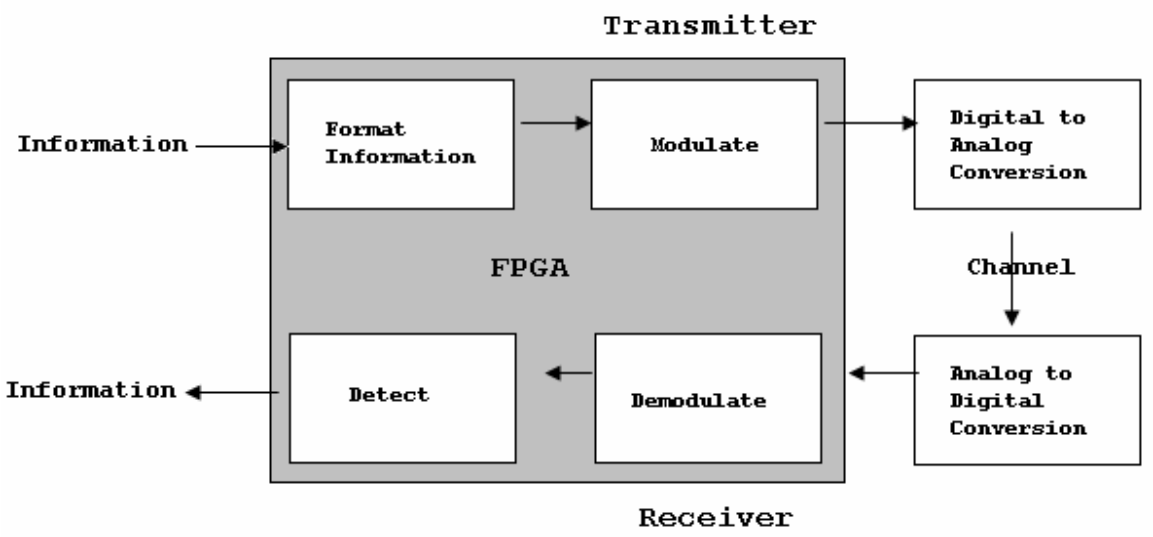

Figure 9. FPGA-RT block diagram.

It is important to note that the design of the RT does not include any filtering, except for that inherent in the receiver's correlator. The initial request from MCTSSA was to program the DSP board with the hardware behavior of the RT-1523C. Figure 9 shows the suggested FPGA-RT block diagram. The channel in the FPGA-RT is a coaxial cable. The filtering inherent in the receiver is ideal for the AWGN case. Non-AWGN environments (e.g. with interferers) might motivate inclusion of additional filtering. Such additional filtering was not addressed in this design.

The design of the FPGA-RT model was approached in a stair-step manner. The first step was to determine the design parameters. Certain design decisions were made based on software restrictions, but most of the parameters were set to facilitate visual analysis and understanding of the system. These parameters are shown in Table 4. For example, assuming the SINCGARS performs orthogonal signaling, then $R_b = 16$ kbps and $\Delta f = 16$ kHz. Therefore the two intermediate frequency (IF) symbol frequencies are

$f_0 = 12.492$ MHz and $f_1 = 12.508$ MHz. If these relatively close frequencies were used, then the visual identification of either waveform would have been difficult. The FPGA includes an 80 MHz crystal oscillator, therefore, this was chosen as the clock frequency [15]. In order to achieve better visual identification of the two IF symbol frequencies, a data frequency of 1.25 Mbps was chosen.

Table 4. Major BFSK-RT Design Parameters.

f_{clock}	80 MHz
f_0	5 MHz
f_1	10 MHz
T_b	800 ns
R_b	1.25 Mbps
BW_{n2n}	7.5 MHz

The second step was to create a transmitter, using the given parameters. To simplify the design, a square wave was designed to simulate an alternating input bit-stream of 0s and 1s. As mentioned before, the Altera DSP Builder can only convert Altera Simulink blocks into VHDL, not MathWorks Simulink blocks. Therefore only Altera Simulink blocks can be implemented in hardware. In the Simulink environment, creating an OOK, FSK or BFSK transmitter is very simple because Simulink has modulation blocks that can perform these tasks. Since the Mathworks Simulink blocks cannot be used and Altera Simulink blocks do not include these functions, the BFSK transmitter had to be created from scratch. The first step taken to create the BFSK transmitter, in order to improve the designer's understanding of the receiver-transmitter (RT) system, was to build an OOK transmitter. [16] [17]

Once the OOK transmitter was built, it was modified to add two OOK signals at different symbol frequencies to produce a BFSK signal, completing the transmitter portion of the RT design.

The implemented receiver is a non-coherent BFSK quadrature receiver. This was designed via a basic Simulink model of a non-coherent BFSK receiver-transmitter that models the RT-1523C modulation using Altera Simulink blocks.

At this point, the Simulink model was input to the DSP Builder design flow using the SignalCompiler. The SignalCompiler then performed three steps: converted the design model to VHDL, synthesized the VHDL and fit the design into the FPGA using Quartus II. The SignalCompiler has a fourth step that programs the DSP board but since the design used an IP, the device programming had to be done from the Quartus II software. After the model was used to create a hardware programming file, the Quartus II software was used to program the FPGA using the tethered OpenCore feature. Once the design was programmed onto the FPGA, hardware testing and evaluation was conducted. [17] [19]

B. ON-OFF KEYING (OOK) TRANSMITTER

One of the basic waveforms used in digital communications is the on-off keying (OOK) waveform. The name is a description of what the signal represents. The presence and absence of a sinusoid pulse denotes a binary one or zero respectively, as shown in Figure 10; thus the ‘on-off’ name. The keying part of the name is a carryover from the days when teletype required a key to send dots and dashes. Since the OOK waveform is easy to create and can easily be

converted into a BFSK waveform, it makes sense to create a working OOK generator that can be modified to create a BFSK transmitter. [22]

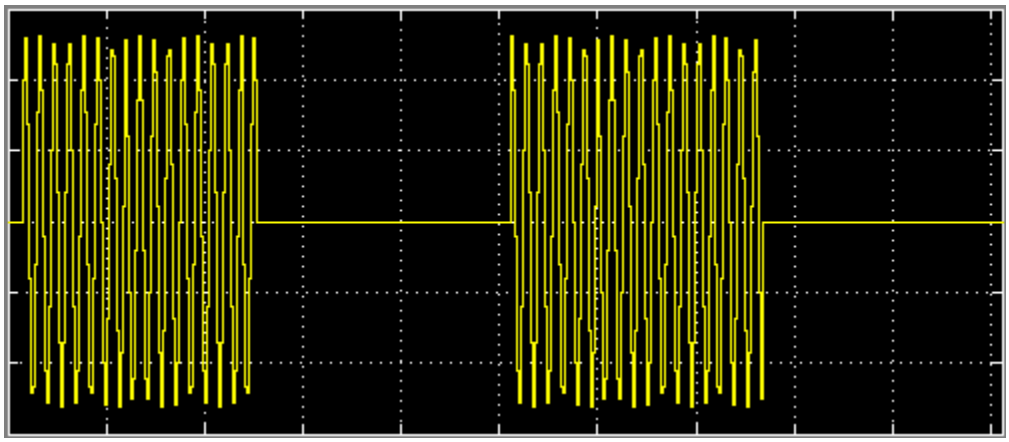

Figure 10. OOK Waveform.

1. Design

An OOK waveform is the product of a sinusoid and the waveform $d(t)$ where $d(t) = \sum_{n=-\infty}^{+\infty} b_n p(t - nT), b_n \in \{0, 1\}$ and $p(t) = \begin{cases} 1 \text{ if } 0 \leq t < T \\ 0 \text{ otherwise} \end{cases}$. The OOK transmitter was constructed using the Altera numerically-controlled oscillator (NCO) MegaCore which creates a sinusoid, an Altera increment block, an Altera extract bit block and an Altera NOT block, which create a square wave, and an Altera product block [17]. Figure 11 shows the model of the OOK dual transmitter that produces complementary OOK signals at 5 MHz and 10 MHz. The signal at point 1 in Figure 11 is a $f_0 = 5$ MHz sinusoid produced by the Altera NCO MegaCore version 2.2.2; the inputs to the NCO are the clock enable, reset and a frequency dependent phase increment constant which the NCO uses to create the desired sinusoid. The signal at point 2 is $1 - d(t)$. The signal at point 3 is $A(1 - d(t))\cos(2\pi f_0 t)$.

The signal at point 4 is $A\left(1-d\left(t\right)\right)\cos\left(2\pi f_0 t\right)+A$. Similarly, the signal at point 5 is $Ad\left(t\right)\cos\left(2\pi f_1 t\right)+A$ where $f_1 = 10$ MHz. The OOK transmitter was designed this way to facilitate the transition to a BFSK transmitter.

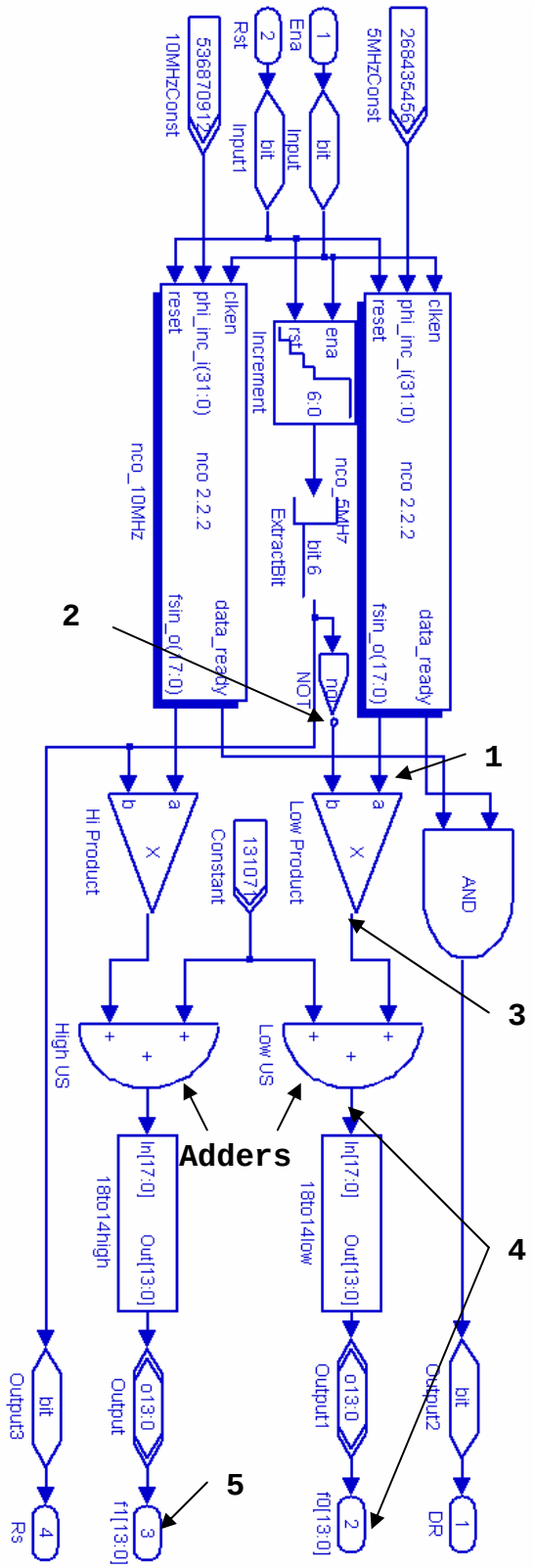

Figure 11. Dual frequency OOK transmitter Model.

One of the key ideas to be noted in Figure 11 is the signed- to unsigned-number conversion done by the adders after the square wave and sinusoid multiplication. This was necessary because of the DSP board DACs. They cannot convert signed numbers to analog [15]. The DAC inputs must be non-negative [15]. Given this restriction, it is necessary to convert the signed values to unsigned values and then reduced in order to successfully transmit a waveform. The signal reduction is performed by 18- to 14-bit busses that remove four bits from the 18-bit signal. This is done by creating an 18- to 14-bit buss that extracts the seven most significant bits (MSBs) and the seven least significant bits (LSBs) from the 18-bit signal and the 14 bits are combined to make a 14-bit signal. Figure 12 shows the 18- to 14-bit model. This approach was selected because removing either the most or least significant bits cause the system to truncate the waveform and produced peaks at the points of discontinuity in the waveform. Removing the bits in the middle of the signal removed these peaks and allowed for smooth sinusoidal signals to be routed to the output.

Another noteworthy feature is the data-ready output (pin 1 in upper right of Figure 11). The NCOs require a certain amount of clock cycles for start-up, and the data-ready signal is asserted once the NCOs are ready to transmit the sinusoids [23]. The data-ready output (pin 1) can be used to debug timing errors.

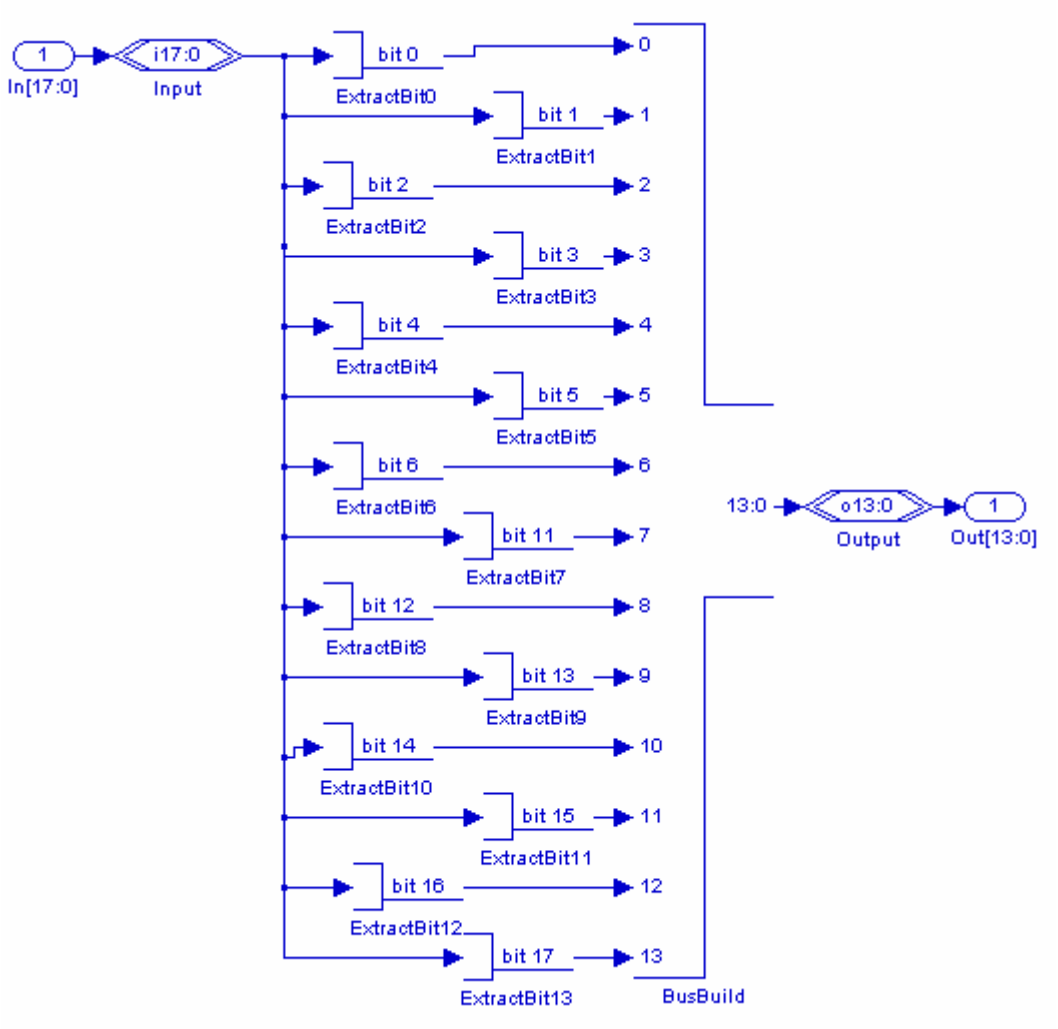

Figure 12. Bus18to14 Model.

a. Numerically Controlled Oscillator

The Altera NCO is an Altera IP which is included with the Altera software [17]. If the designer finds it useful, a license maybe purchased from the Altera website and the MegaCore can be used in hardware implementations and products. The NCO requires the use of the Altera SignalCompiler block in order to create, compile, and operate the oscillator. Once the SignalCompiler has been added to the top-level of the model, any of the Altera MegaCores can be used in the model. [19]

The NCO parameters are established using the Altera NCO Compiler, a graphic user interface (GUI) that allows parameters to be set for the NCO. The Parameterize window has three tabs: *Parameters*, *Implementation* and *Resource Estimate*. The *Parameters* and *Implementation* tabs are the most important because they determine the specific behavior of the NCO. The *Resource Estimate* tab allows the designer to note how many of the FPGA resources the NCO may use. Figure 13 shows the *Parameters* tab of the NCO compiler. [23]

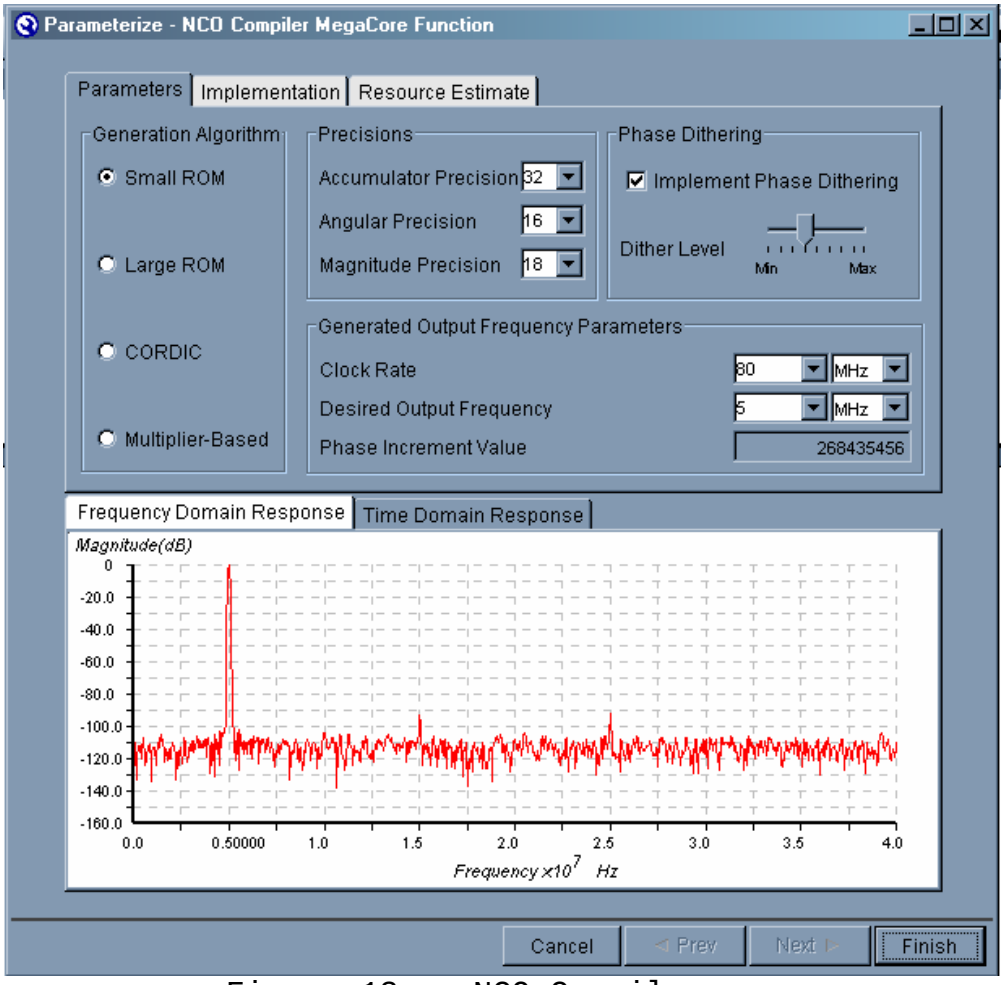

Figure 13. NCO Compiler.

The most important part of the compiler is the *Generated Output Frequency Parameters* section in the *Parameters* tab (see Figure 13). In this section the *Clock Rate* and *Desired Output Frequency* are set. [23]

b. Square Wave Implementation

The technique used to create the bit stream was to implement a square wave with a period equal to twice the bit duration. Each square wave period represents two bits, a 0 and a 1. Since the Altera blockset does not contain a square wave generator, one was created by using an increment block and an extract bit block [17]. The increment block is a counter. The extract bit block extracts the counter's MSB.

The increment block was used due to timing issues. The design uses a global sampling time. This was done to reduce the chances of timing errors or delays that in turn could cause synthesis errors. This issue is discussed more in depth in Appendix A.

The square wave was implemented by extracting the MSB of an up counter. This caused the square wave to change from 0 to 1 once the counter reached half its maximum value. Given this, the bit rate was determined by the size of the counter. In Figure 14, the increment block acts as a 7-bit counter that can count up to 128 therefore the bit rate (R_b) is 1 bit per 64 clock cycles or 1.25 MHz.

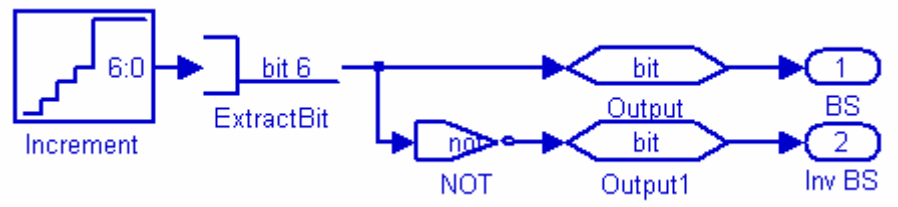

Figure 14. Square Wave Source.

In order for a Simulink model design to be converted to VHDL, the model has to establish the input and output signals using input and output busses. The input and output ports, i.e., the ovals with numbers inside them seen in Figure 14, allow models to be made into subsystems that can be used a part of a larger model. In order to use a subsystem in the Signal Compiler, the subsystem has to have a "SubSystem AlteraBlockSet" mask. Figure 15 shows the OOK dual transmitter top level model, including a modulator mask representing the subsystem of Figure 11. [17]

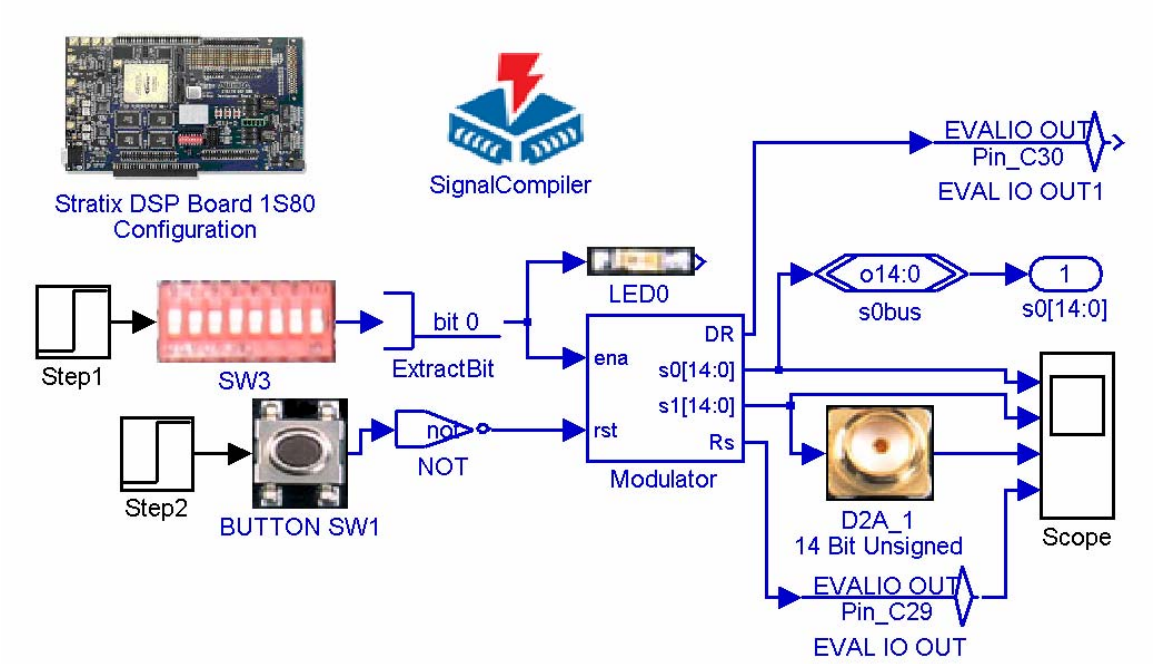

Figure 15. Top Level OOK Transmitter Model.

C. BINARY-FREQUENCY-SHIFT-KEYING (BFSK) TRANSMITTER

A binary-frequency-shift-keying (BFSK) signal is a representation of 1s and 0s using sinusoidal pulses at two different frequencies as represented in Figure 16 [22]. Given this fact, the conclusion can be drawn that the com-

bination of two complimentary OOK signals with distinct tone frequencies creates a BFSK signal.

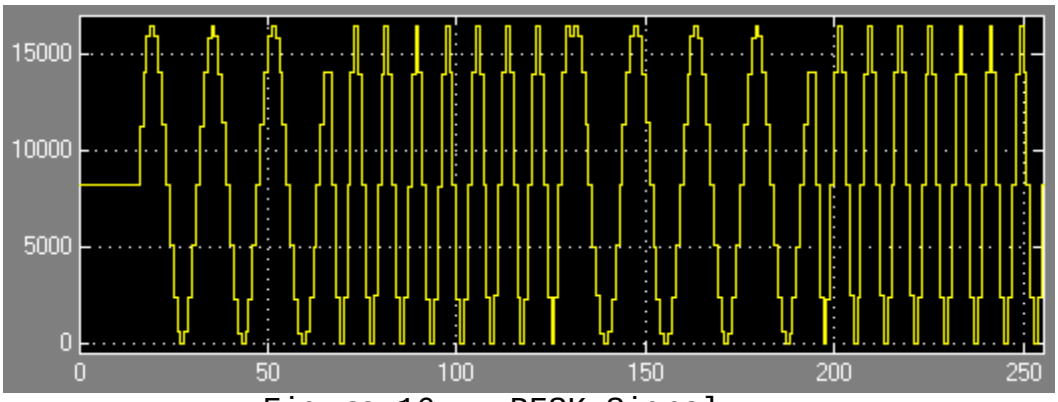

Figure 16. BFSK Signal.

By examination, it is intuitive to simply add the two waveforms to create a BFSK waveform. Figure 17 shows the model of the BFSK transmitter. The signals at points A and B are the complementary OOK signals. The sum signal at C is the BFSK signal. The signal at D is the unipolar BFSK signal.

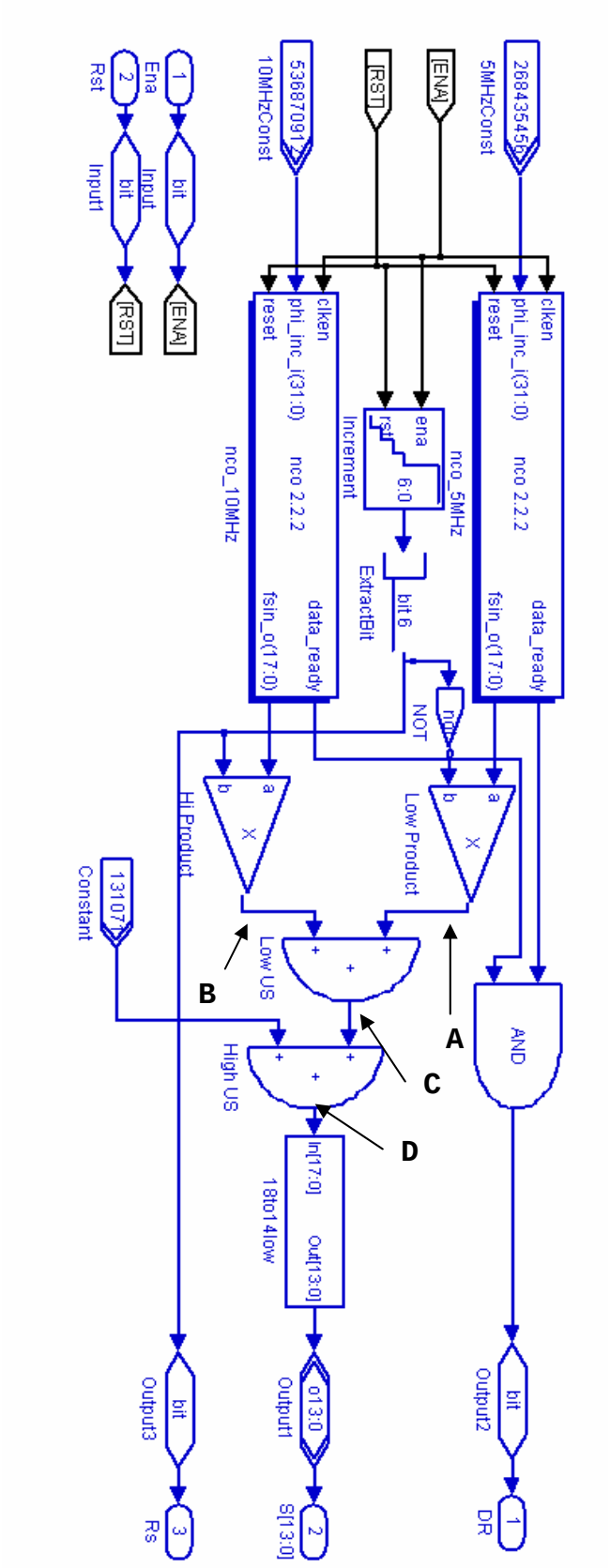

Figure 17. BFSK Transmitter Model.

D. NON-COHERENT BFSK-RT

The purpose of a non-coherent receiver is to evaluate which sinusoid is stronger at the receiver at a given time without using the carrier phase to detect the signals. The two most common non-coherent BFSK receiver designs are the energy detector, also known as the quadrature receiver, and the envelope detector. For this thesis, the quadrature receiver was implemented because of its simplicity and low cost. [21]

The BFSK quadrature receiver can be implemented as seen in Figure 18 and operation of a BFSK quadrature receiver is discussed in more detail in Reference 21.

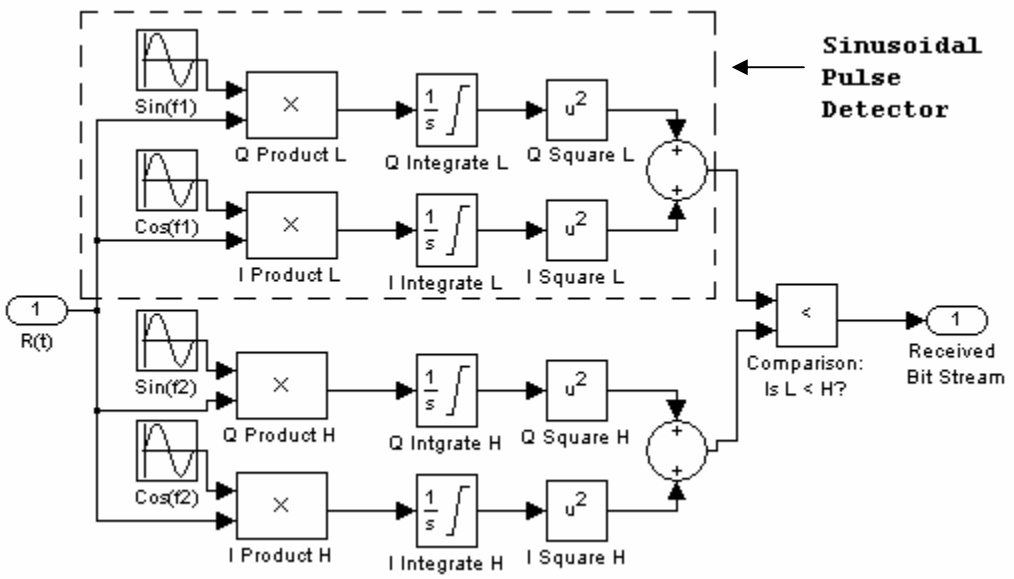

Figure 18. Non-coherent BFSK Receiver (from [21]).

1. Design

The conceptual block diagram was implemented using Altera Simulink blocks to make sinusoidal pulse detectors as seen in Figure 19. Two instances of the sinusoidal pulse detector were used to make the non-coherent BFSK receiver.

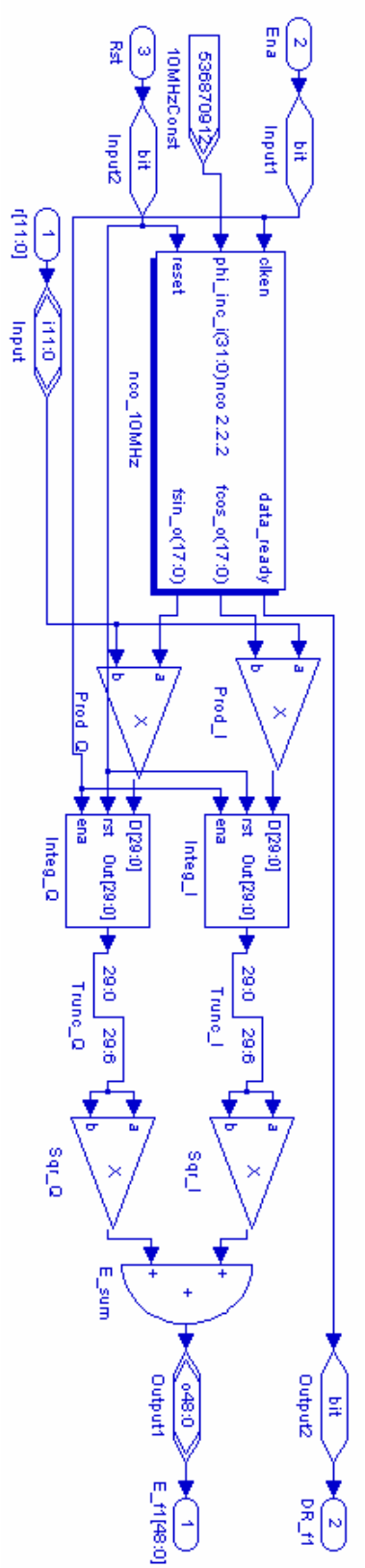

Figure 19. Sinusoidal Pulse Detector.

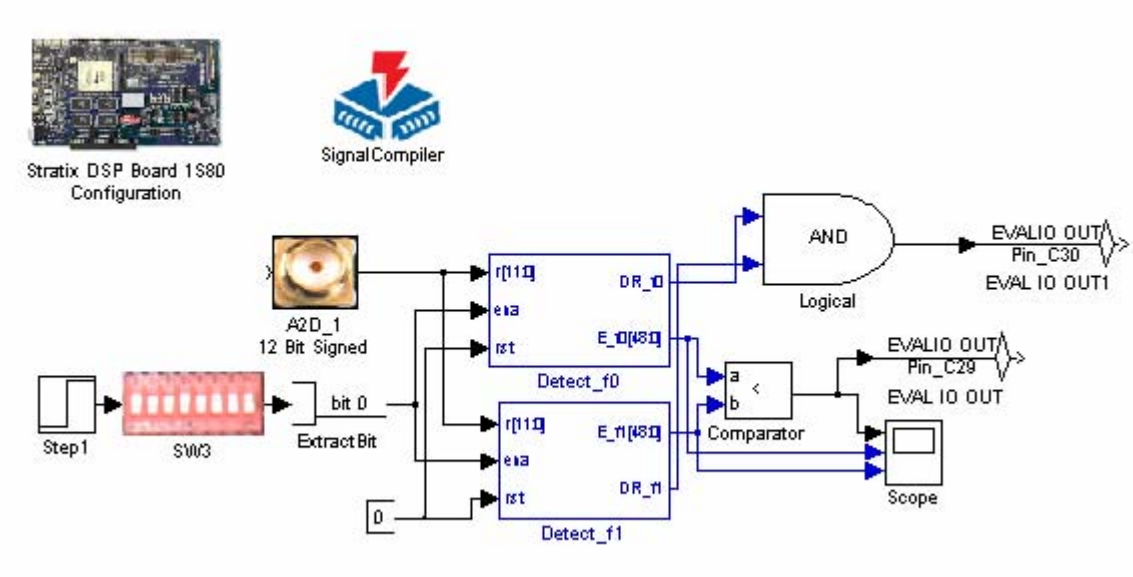

Figure 20. Non-coherent BFSK Receiver.

The last step of the design was to combine the transmitter and receiver into one BFSK-RT design. The action of combining the models was a simple cut and paste operation that combined the models from Figure 20 and Figure 17 to create Figure 21. In software simulation, this designed worked properly and displayed no problems. This was not the case in hardware, and will be discussed in the BFSK-RT sub-section of the HARDWARE IMPLEMENTATION section.

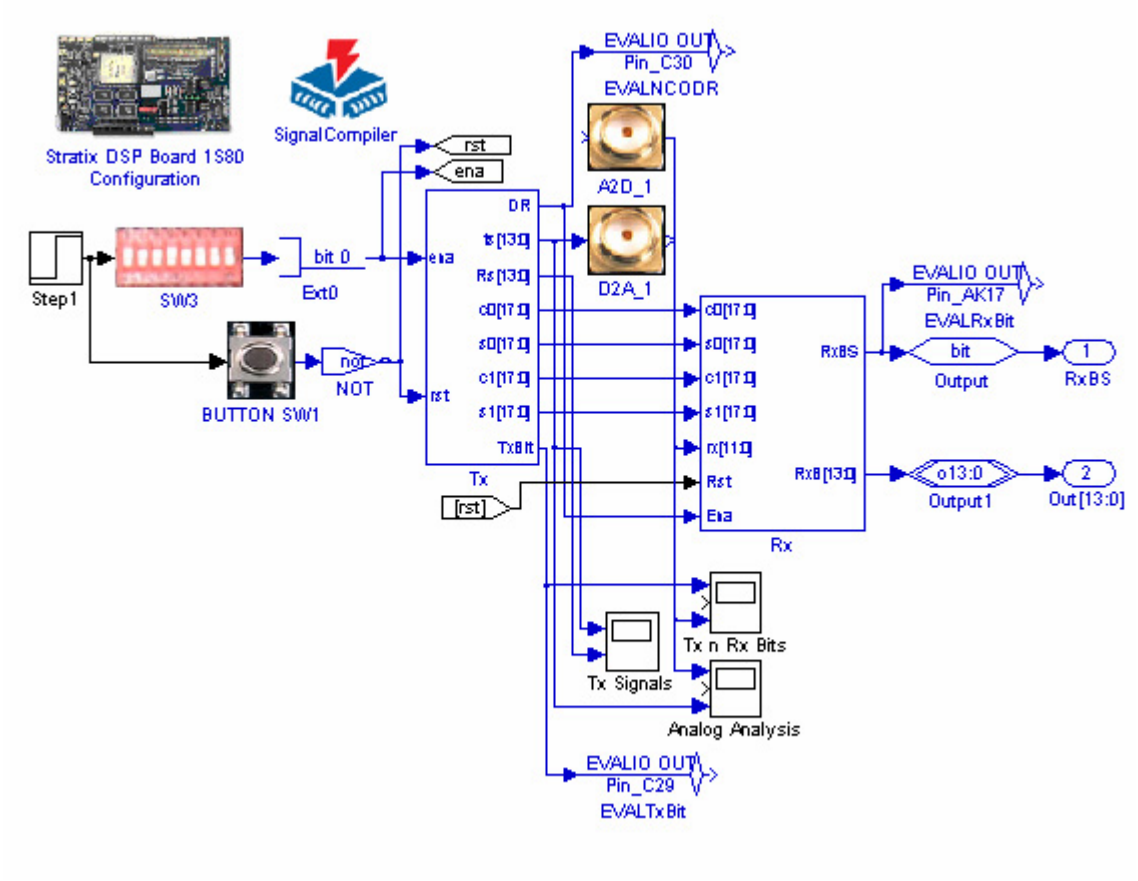

Figure 21. BFSK-RT Model.

F. HARDWARE IMPLEMENTATION

The hardware implementation became the most challenging step of the design process, mostly due to lack of knowledge of software and IP intricacies. Every design was implemented into hardware, mostly as a troubleshooting technique, to validate the design. The observations from each of the hardware implementations will be presented in the Results chapter. The following sections will discuss the steps taken to implement the designs. Errors and lessons learned can be found in Appendix A.

1. Required Altera Blocks

It was stated earlier that in order to use any of the IPs, it is necessary to include a SignalCompiler block in the top-level model. Similarly, to program the DSP board,

the DSP-Board block has to be added to the top-level model. The Altera DSP-Board library also includes multiple blocks that represent the features available on the DSP board. For example, in Figure 21, there are two gold circles that represent the ADC and DAC and two EVALIO OUT blocks that assign certain signals to IO pins on the board. [17]

Some of these blocks were observed to have no functionality in the Simulink simulator, like the push buttons and the dip switches, but they are necessary in the design in order to power and control the system in hardware. Other blocks were observed to have effects on the simulation, for example the ADC and DAC blocks modify the signal if the signal routed through them is not in the proper format.

The most important block for programming the system onto the board is the board block. When the block is double clicked the board configuration window is displayed, Figure 22, which allows the designer to set certain board parameters. If this window is not modified, the default configuration will be used. The default configuration does not set a clock for the DACs, therefore the DACs will not transmit any signals out of the board. An additional parameter that can be set from the board configuration window is the global reset pin. If the global reset is not set, Quartus II selects any free pin and the system cannot be reset by the user. In this implementation, the global reset used was pin AK13 which is push button 1. This allowed for user control of the system reset during hardware evaluation. [17]

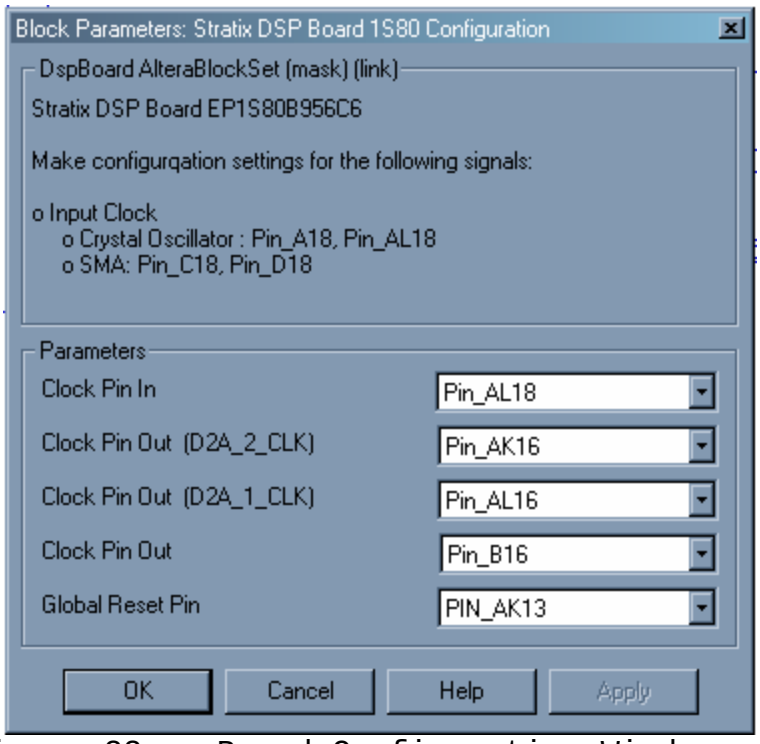

Figure 22. Board Configuration Window.

2. DSP Board Programming

Once the necessary blocks were in place to convert the model to VHDL, the programming of the board was started. The programming started with a counter implementation to ensure the input and output ports were working properly, and to verify that the timing of the system was as expected. Once a working model was programmed and the functionality and use of the board was established, the three designs were implemented: the OOK transmitter, the BFSK transmitter and the BFSK-RT. Figure 23 shows a picture of the hardware set-up.

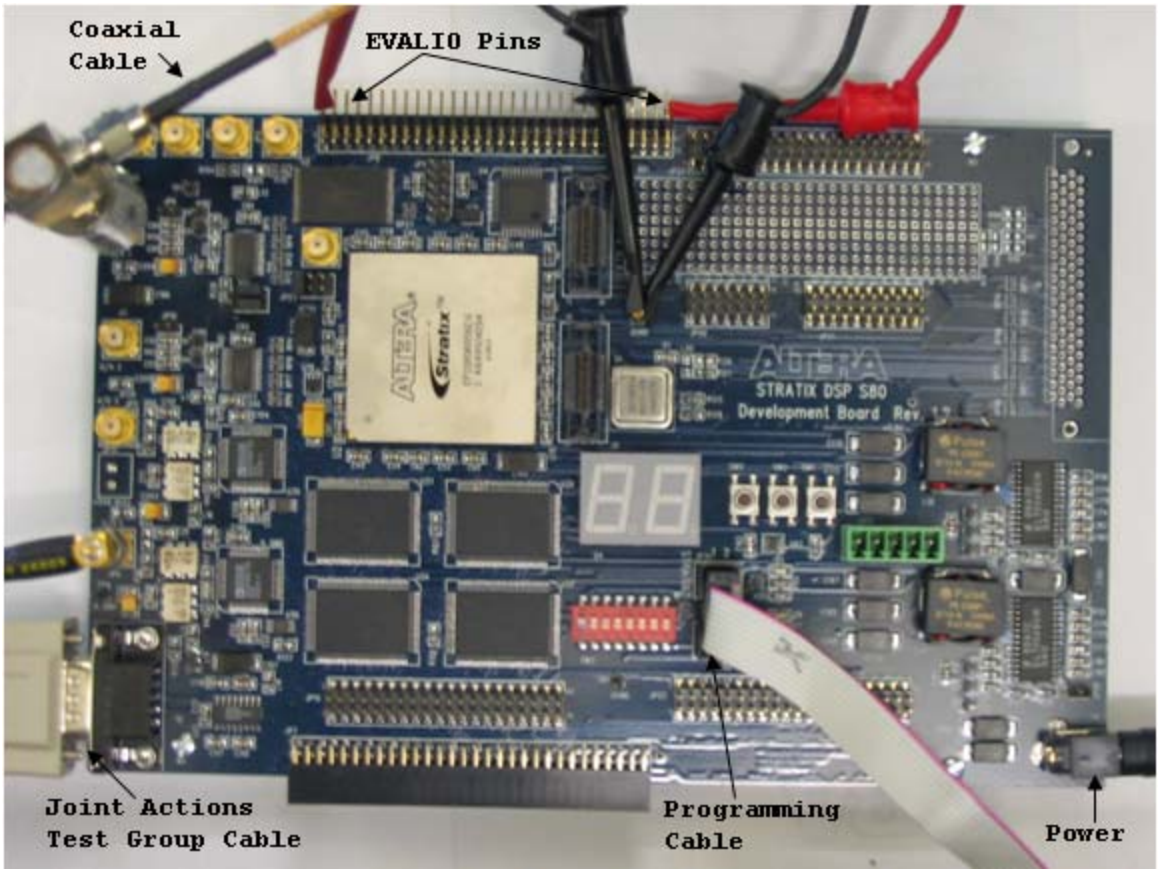

Figure 23. DSP Board Set-Up.

a. 8-Bit Counter

The first design to be programmed onto the board was an 8-bit counter. This exercise provided an example of how the board uses the clock to determine frequencies. Using equation: $f_{counter} = \frac{f_{clock}}{2^N} = \frac{80\ MHz}{2^N}$, from Appendix A, yields a frequency of 312.5 kHz for the 8-bit counter because it uses the onboard crystal oscillator which operates at 80 MHz. When the MSB is extracted from an 8-bit counter, the extracted bit output creates a square wave with a frequency of 312.5 kHz, the same frequency as the counter. The counter operation displayed the expected frequency output, and ensured the parameters of the design were correct. The output oscilloscope screen capture can be seen in Figure 24 below. Although the oscilloscope

measured frequency for the counter output, top waveform, is incorrect, the correct frequency can be calculated from the period using the equation $f = \frac{1}{T}$, where $T = 3.2\mu s$ [22].

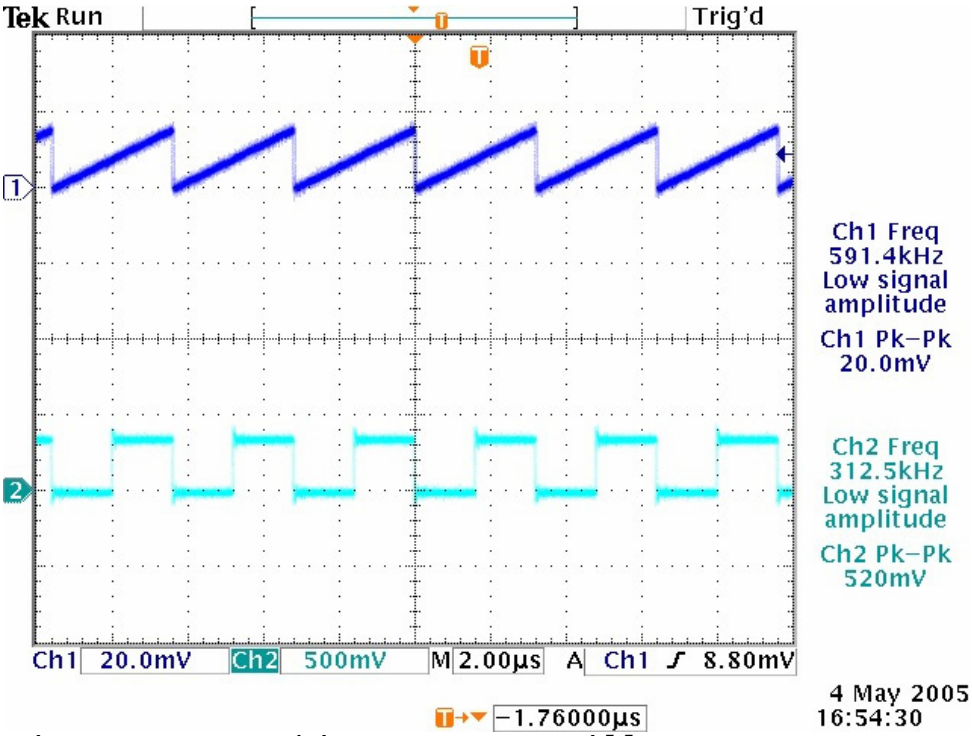

Figure 24. 8-bit Counter Oscilloscope Output.

b. OOK Transmitter

Once the board was determined to be working properly, the OOK transmitter design as shown in Figure 11 was implemented into hardware. The OOK implementation provided some good learning points regarding the use of the dip switches and the push buttons. All the board switch inputs are low-asserted, therefore they drive a logic-0 when on and a logic-1 when off [15]. Using these characteristics, a control arrangement was established and used through out the design. Since the push buttons drive a logic 0 only when pressed, push button one (SW1) in combination with a NOT block was used as the system reset. The dip switch was

used as the system enable, which allows the system to be turned on and off at any given time.

c. BFSK Transmitter

After the OOK transmitter was implemented and tested, the BFSK transmitter was implemented as a trouble-shooting step. This implementation was completed without any errors, since the only difference between the BFSK and OOK transmitters is the rearrangement of the adders to add the high- and low-frequency signals onto the same waveform and then convert them into unsigned-integer values.

d. BFSK-RT

The BFSK-RT hardware implementation proved to be one of the most challenging implementations due to design structure. The design approach was to use each design to build the next design. In software, this approach was able to simulate without any problems. The hardware implementation did not encounter any problems until the transmitter and receiver models were combined. In software simulation, Simulink does not give an error when two sources have the same exact parameters in one design. This is not the case in hardware. The SignalCompiler would not compile two NCOs with the same parameters. Therefore, the design was altered to include two NCOs instead of four, as pictured in Figure 21, thereby eliminating redundant NCOs.

This chapter discussed the design steps taken to implement a BFSK-RT into an FPGA. The observed results are presented in the following chapter.

THIS PAGE INTENTIONALLY LEFT BLANK

IV. RESULTS

This chapter provides a summary of the results of the simulations and the hardware implementations throughout the design process. The chapter is broken down into two main sections, and each section discusses a major design task and the results of the software and hardware implementations.

A. SOFTWARE IMPLEMENTATION

Throughout the entire design process, simulations and evaluation of the design were made to ensure the steps taken were correct and would lead to a working BFSK-RT. This section discusses the observed behavior of the OOK transmitter, the BFSK transmitter and the BFSK receiver and how the intermediate observations affected the final design.

Although the Altera software contains multiple analysis tools, the only software analysis tool that was used was the Simulink simulator. The Simulink simulator and its scope block provided the most basic and easy-to-understand analysis method that could be performed without having to switch software or incorporate hardware for testing. [24]

Since a receiver-transmitter radio set must first be able to send and receive a waveform, the analysis of the output waveform was the method used to ensure the design worked properly. This was done by using the Simulink scope block. The scope block receives a signal and plots it against the time index. The output of the scope is what would be expected on the screen of an oscilloscope if it where to be used to acquire a single sweep of a waveform. The only difference is that in the Simulink environment,

the single sequence size is set by the user as the total simulation time. Using this voltage-against-time plot, the design could be inspected to ensure the system was operating as expected. Table 5 lists the parameters used for all the software simulations. [24]

Table 5. Simulation Parameters.

f_{clock}	80 MHz
f_0	5 MHz
f_1	10 MHz
Simulation Time	0 to 350
Sample Time	1 sample (12.5 ns)
Symbol Time (T_s)	64 samples
Symbol Rate (R_s)	1.25 MSPS
BW_{n2n}	7.5 MHz
NCO Phase Dithering	Level 4
NCO implementation	Small ROM

1. OOK Transmitter

Upon completion of the OOK transmitter, an analysis of the waveform was performed. As mentioned before, an OOK transmitter performs modulation by transmitting a signal to represent a binary one and no signal to represent a binary zero [22]. Figure 25 shows the system output waveforms and the transmitted bit stream of the created OOK transmitter.

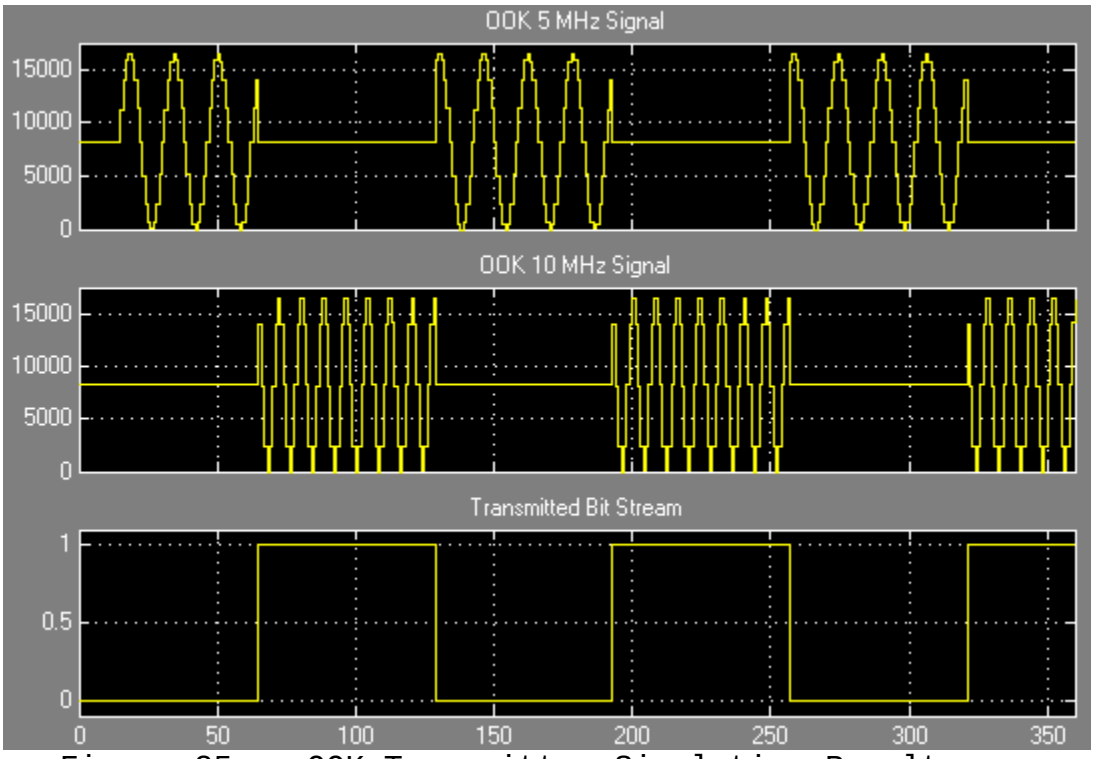

Figure 25. OOK Transmitter Simulation Results.

By simple visual comparison, it can be noted that the waveforms behave properly, the 10 MHz signal is present only during transmitted ones and the 5 MHz signal is present when the data transmitted is a zero [22]. Another significant observation is the frequency of the waveforms. The 10 MHz waveform has twice as many cycles as the 5 MHz waveform, as would be expected. It was during this step in the design process that the NCO parameters were explored and tested. The results of these iterations led to the decision to use the 5 and 10 MHz frequencies with a clock frequency of 80 MHz. These parameters provided clear and identifiable waveforms. Additional parameters including phase dithering and implementation algorithm were also tested and examined and it was determined that the phase dithering affects the frequency response of the waveform and the implementation algorithm affects the amount and

type of resources used by the FPGA. The small ROM method uses the least logic elements, but the most memory resources. Further details on resource usage based on implementation algorithm may be found in the Stratix Device Handbook, Reference 14. [23]

2. BFSK Transmitter

The BFSK transmitter simulation produced a distinct binary waveform that alternated the 5 MHz and 10 MHz waveforms into one signal. Figure 26 displays the resulting BFSK waveform, along with the corresponding bit stream.

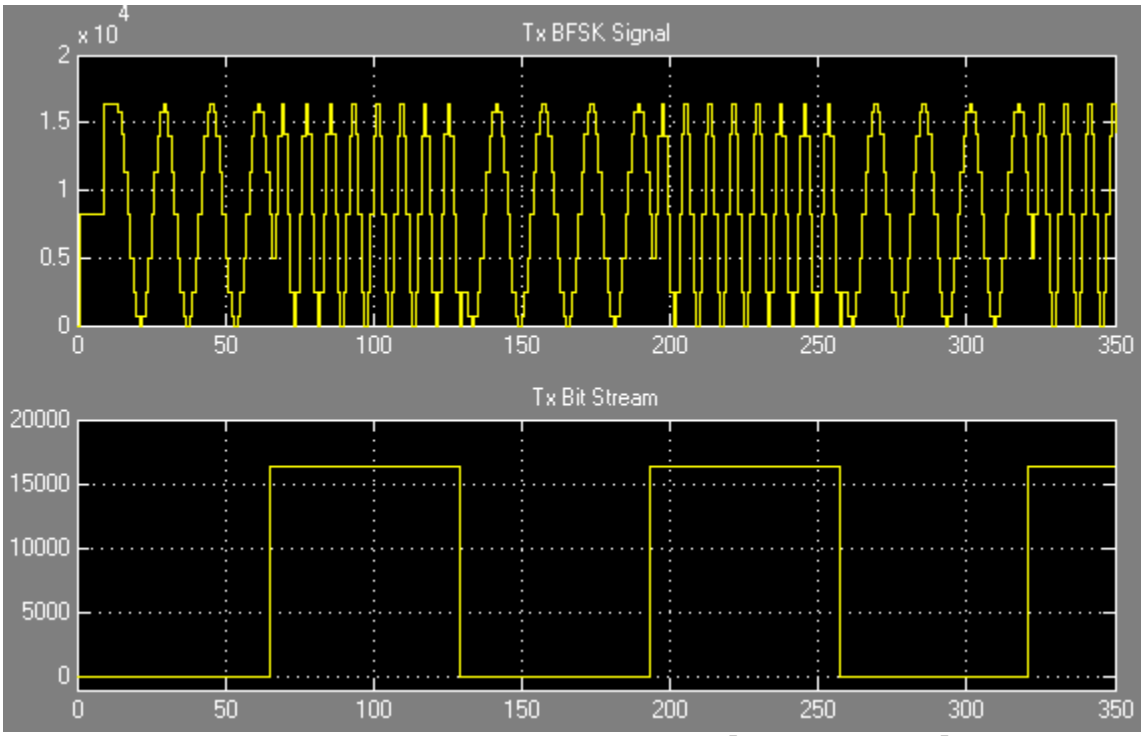

Figure 26. BFSK Transmitter Simulation Results.

It was during this simulation that the effects of pipelining, adding a memory element to the output of a function to reduce timing errors, were noticed. The implemented transmitter design incorporates pipelining after all major arithmetic functions; the NCOs, the multipliers after the NCOs, the adder that combines the signals and the adder

that converts the AC waveform into a DC waveform. The pipelining incorporates an overall delay of four clock cycles, which was noted when comparing the transmitted bit stream and received bit stream in Figure 27. The transmitted waveform has been shifted by four samples, as would be expected since each level of pipelining adds a delay to the data.

3. BFSK-RT

The BFSK-RT implementation provided satisfactory results when simulated. The transmitted waveform was the same as seen in Figure 26, and the behavior of the BFSK-RT can be seen in Figure 27.

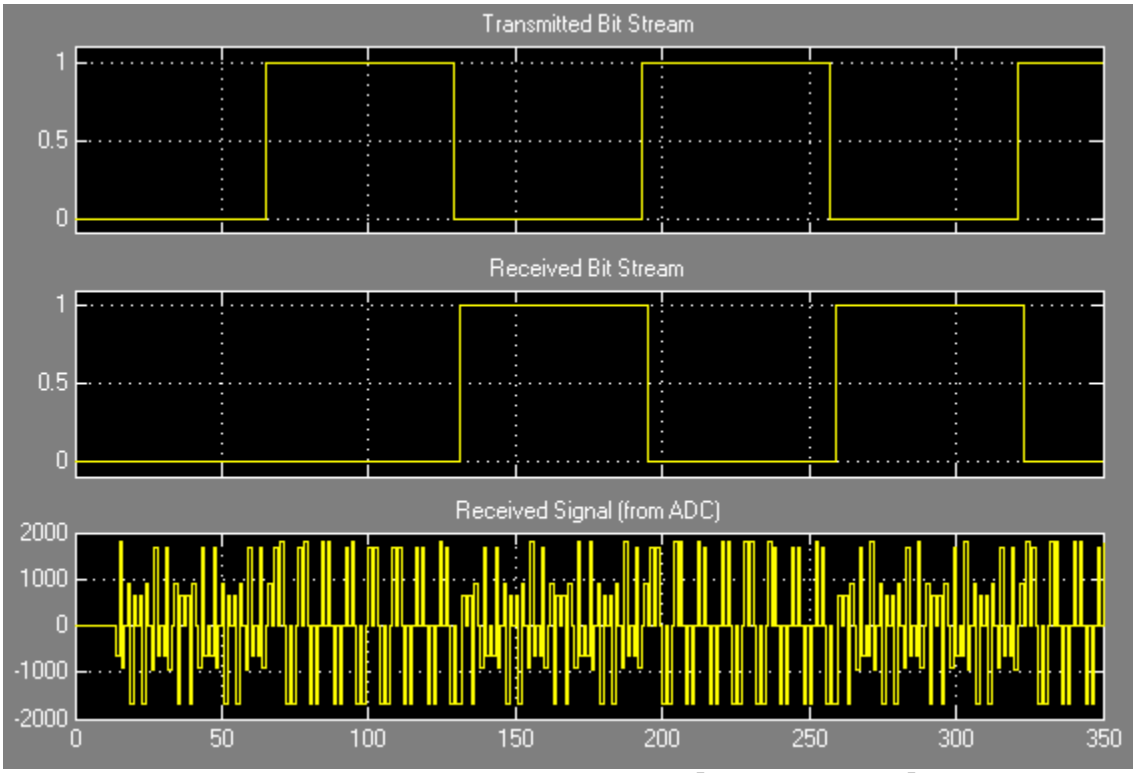

Figure 27. BFSK Receiver Simulation Results.

By visual examination of Figure 27, it can be noted that the receiver has a one bit delay, 64 samples, creating a shifted received waveform. This behavior is expected be-

cause the integrators in the detection stage of the receiver integrate over a bit period. At system start-up, the integrators have no data to determine the data received until a bit period has passed. After the first bit period has passed, the integrators determine what data was received and the receiver can then detect the value of the received signal. This configuration forces the transmitted and received waveforms to be shifted by T_b, or one bit.

One of the most interesting observations of the BFSK-RT simulation was the received signal appearance. As can be noted from Figure 27, the receiver system input from the ADC is not a BFSK waveform. The waveform should be a two's-complement sinusoid with an amplitude of 2^{11}, or 2048, because the MSB is the sign bit. The ADC should not detect a value, but rather a waveform which it converts to a 12-bit digital value. Instead, the ADC block acts as a 12-bit signed integer bus that causes the ADC to truncate the input to 12 bits, therefore the signal is improperly displayed in the scope. This discrepancy was examined in the hardware to determine if the block implementation in the Altera library was incorrect or if the ADC actually input the observed waveform into the receiver, and it was determined that the system received the proper waveform.

B. HARDWARE IMPLEMENTATION

Upon successfully designing the BFSK-RT, the task of programming the design onto the FPGA was started. In order to get familiar with the board and the programmer, each design was implemented and tested on the FPGA. The following sections discuss the observations made during the hardware testing. The analysis was done using the Tektronix TDS3012B two-channel color digital phosphor oscilloscope.

1. OOK Transmitter

Figure 28 below shows the oscilloscope capture of the OOK waveform and the transmitted bit stream. It can be noted that the OOK signal behaves exactly as expected, with a sinusoid present when a binary-1 is transmitted and no sinusoid present when a binary-0 is transmitted.

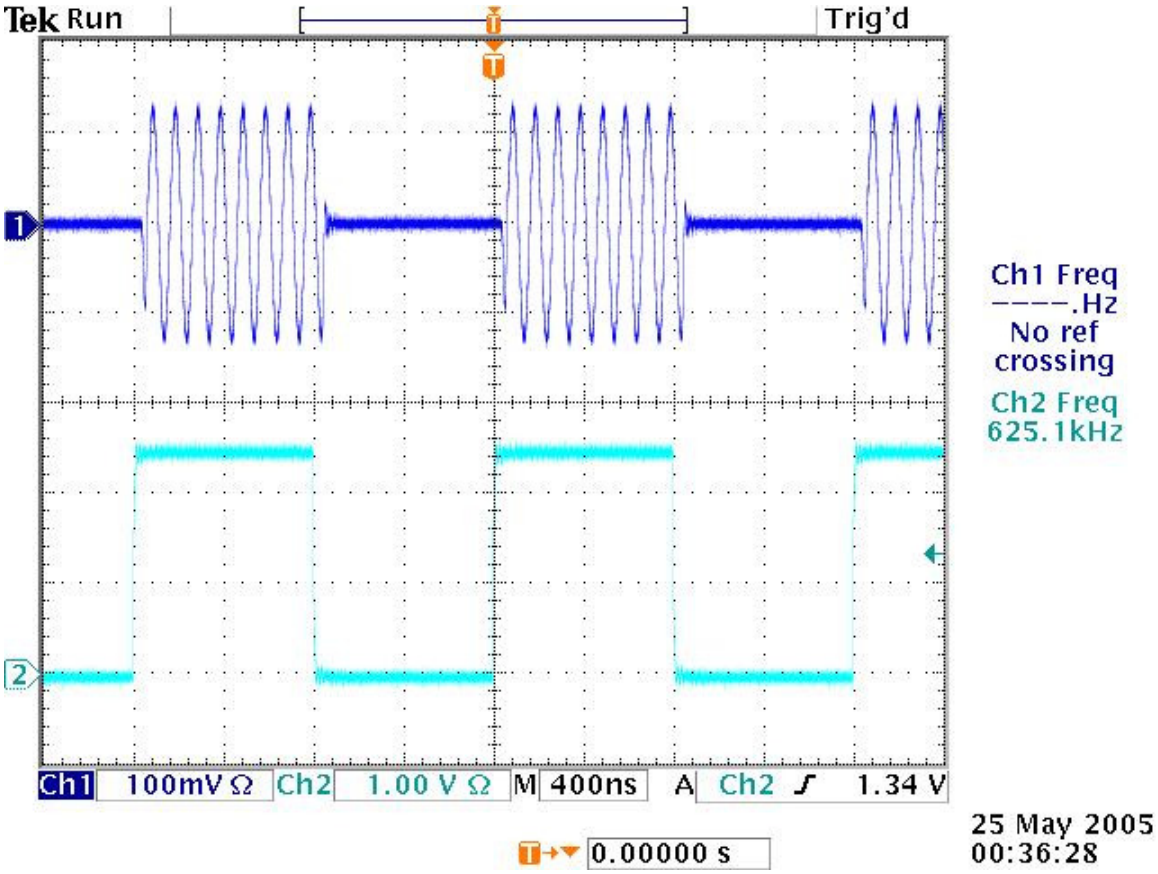

Figure 28. OOK Waveform and Bit Stream.

One noteworthy evaluation tool implemented in the OOK transmitter and used through out the hardware implementations was the use of the DSP board IO pins. Since the DACs convert 14-bit data to analog signals, a one-bit output contained insignificant power and was lost in the noise. To alleviate that, the IO pins were used to analyze the bit streams and not waste the DACs on one-bit outputs. The IO

pin output is 3.3 V for a “1” and 0 V for a “1”, which is easily read on an oscilloscope. [15]

2. BFSK Transmitter

Since the OOK transmitter hardware implementation was uneventful, the BFSK transmitter implementation was easily programmed and evaluated. Figure 29 below shows the observed waveforms.

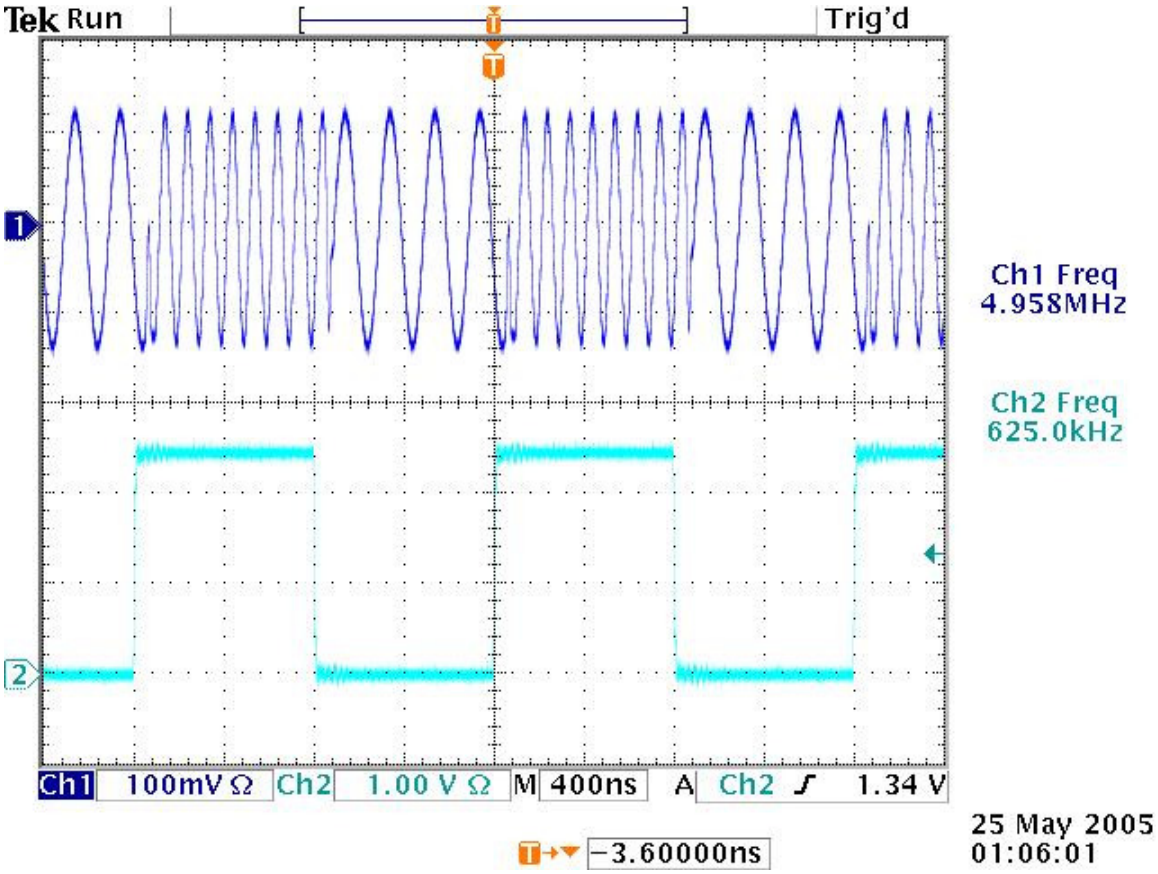

Figure 29. BFSK Waveform and Bit Stream.

As expected, the waveform has a 10 MHz symbol when a binary-1 is transmitted and a 5 MHz symbol when a binary-0 is transmitted. This is also the first time a distinct time delay can be noticed in the waveform. The actual delay in the transmitter system is approximately 62.5 ns which is attributed to the pipelining which adds 4 samples

or 50 ns to the transmitter delay and electromagnetic propagation. This was validated when the oscilloscope capture is zoomed to a 40 ns scale and the waveform and bit stream time differences were compared. [21]

3. BFSK Receiver-Transmitter

The overall design of the BFSK-RT is a resource inexpensive design, as Table 6 shows. Important benchmarks are the LE use at 2%, memory bits at 7%, M4Ks at 37%, and DSP block use at 22%, which means the device still has plenty of resources to incorporate more functions to the BFSK-RT so it can be modified to more effectively simulate the RT-1523C. [14]

Table 6. Quartus II Fitter Report.

Total logic elements 1,721 / 79,040 (2 %)
Total LABs 209 / 7,904 (2 %)
Logic elements in carry chains 617
User inserted logic elements 0
Virtual pins 0
I/O pins 57 / 692 (8 %)
Clock pins 1 / 20 (5 %)
Global signals 13
M512s 0 / 767 (0 %)
M4Ks 136 / 364 (37 %)
M-RAMs 0 / 9 (0 %)
Total memory bits 557,056 / 7,427,520 (7 %)
Total RAM block bits 626,688 / 7,427,520 (8 %)
DSP block 9-bit elements 40 / 176 (22 %)
Regional clocks 0 / 16 (0 %)
Fast regional clocks 0 / 32 (0 %)
SERDES transmitters 0 / 152 (0 %)
SERDES receivers 0 / 152 (0 %)
Maximum fan-out node clock
Maximum fan-out 1110
Total fan-out 10537
Average fan-out 5.40

The final design consists of a transmitter that sends a continuous stream of alternating ones and zeroes that are then transmitted from the DAC via a SMA cable to the ADC, the receiver input. The received waveform is then processed through the non-coherent quadrature receiver and the

system output is the received bit stream. Figure 30 shows the sent and the received BFSK waveforms.

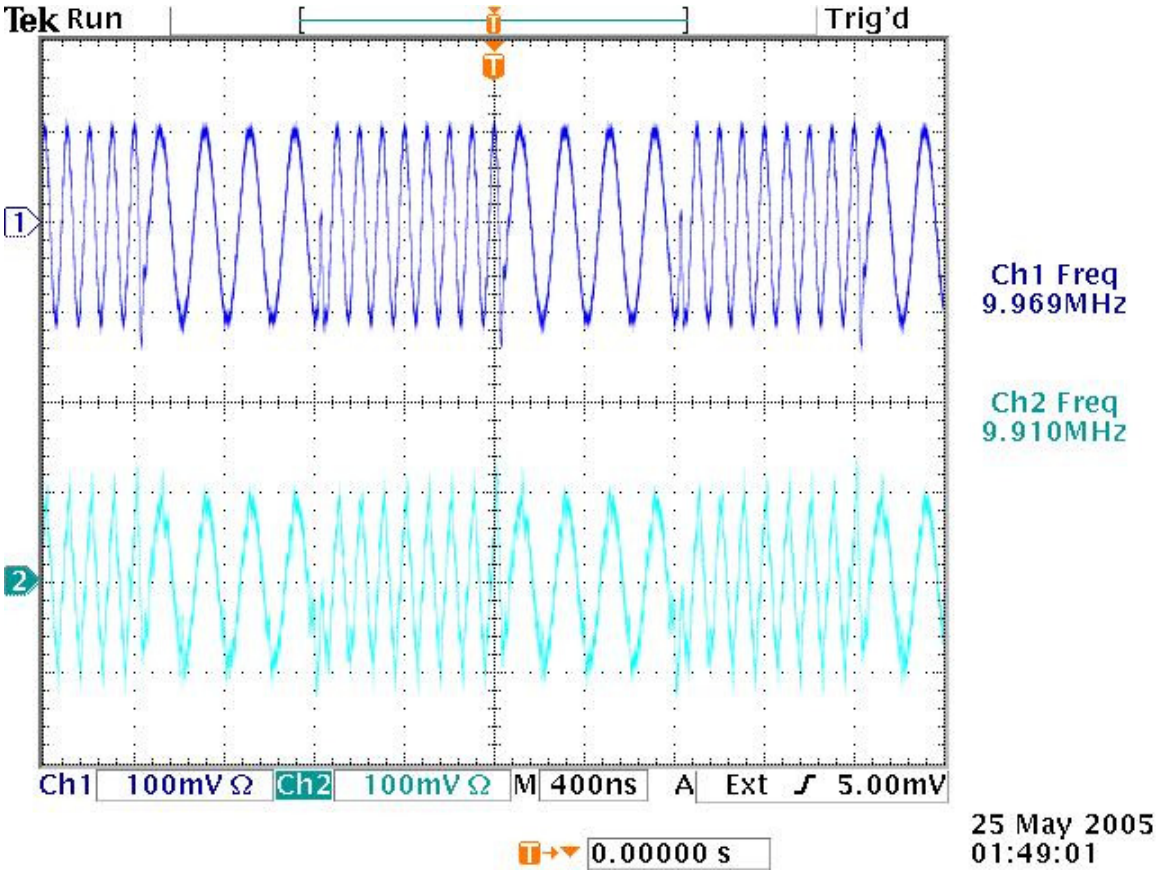

Figure 30. BFSK-RT Transmitted and Received Waveforms.

Since the sent and received signals are essentially the same signal just observed at the output of the transmitter and the input of the receiver, the delay is negligible. When observed in the oscilloscope at a scale of 10 ns, the delay from sent to received waveform is approximately 15 ns. This delay can be attributed to electromagnetic propagation. Figure 31 below shows the oscilloscope display of the sent and received bit streams.

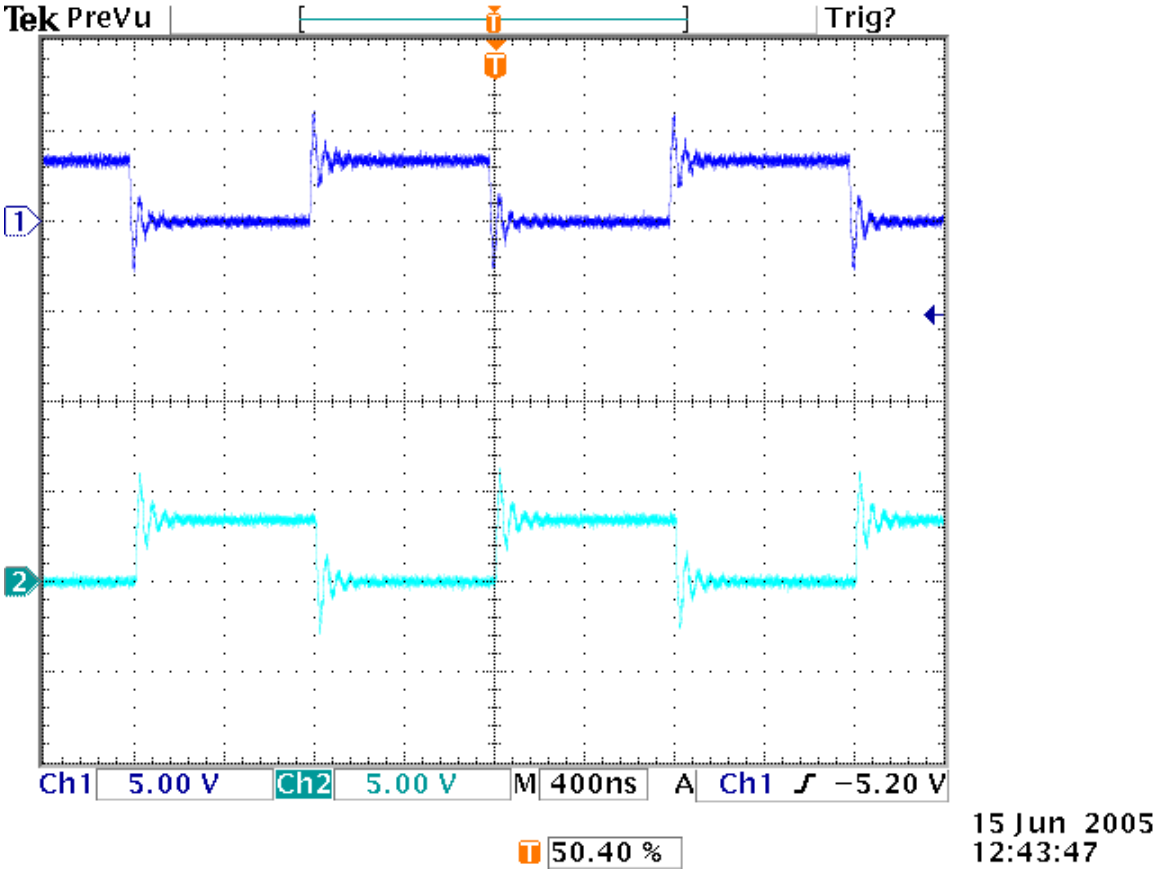

Figure 31. BFSK-RT Transmitted and Received bit streams.

As expected, the bit streams have the same data rate of 1.25 MHz and the received bit delay is one full T_b, 800 ns, as Figure 31 shows.

C. COMPARISONS

Due to the distinct difference between software and hardware evaluation, it is worthwhile to annotate any differences between the two analyses. The software simulations were used for design evaluation and assistance in determining the expected behavior of the system. After the system behavior was determined, it facilitated identifying possible problems in hardware implementation. This approach allowed for the successful implementation of all the designs and for the identification of various errors which are discussed in Appendix A.

The major difference in the two simulation methods was the BFSK waveform appearance at the receiver/ADC. The hardware showed that the sent and received BFSK waveforms had the same appearance and frequencies, whereas the software simulation resulted in a received waveform that was a truncated signal and not a sinusoid. After software examination, it was determined that the problem was not the hardware ADC, but rather the software Altera ADC block. In software the signal needs to be routed through the Altera DAC and ADC blocks. The ADC block expects the signal routed through it be a 12-bit signed integer or an 11-bit unsigned integer, it cannot take a waveform (a signal that is analog in real systems) and convert it to a 12-bit two's complement digital value. Instead, the ADC acts as a 12-bit bus. This behavior causes the received signal to have the incorrect appearance, although the ADC and DAC perform the proper operations in hardware. [15]

The two analysis methods in combination provided a good way to create and analyze the FPGA BFSK-RT, and provided various possibilities for continued work and further analysis as will be discussed in the next chapter.

V. CONCLUSION

This chapter provides a summary of the thesis and draws conclusions regarding the BFSK transmitter-receiver implemented in an FPGA. Suggestions and recommendations are given regarding further research and changes to the design created.

A. SUMMARY

The goal of this thesis was to provide the first step towards developing a solution to the data transfer problem in the current manpack version of the SINCGARS. MCTSSA approached NPS with a request to create and implement the current manpack radio, the RT-1523C in an FPGA. In response, this thesis has implemented a non-coherent BFSK receiver-transmitter system in an FPGA.

This thesis gives a thorough description of FPGAs, the Altera DSP development board which has an Altera Stratix FPGA, and how the FPGA design software works. Once the basis for FPGA design was presented, the task of designing the RT was described step by step including noted software and hardware issues that affected the design process. The system was then simulated and analyzed both in software and hardware to ensure proper functionality. The results are presented in the Results chapter.

B. CONCLUSIONS

The most significant conclusion is that the Altera development board is a very capable design tool that can successfully be used to implement a non-coherent BFSK system. The free-trial method of using the Altera IPs allows designers to program devices and test them in hardware at no

additional cost. Also noteworthy is the large amount of design, testing, and debugging options available on the board.

The fact that the board was able to hold the design with small resource usage means that the board can be used to implement much more than the modulation. The only resource that was more than 10% used was the M4K memory blocks at 37%. The board can easily hold the coding and even the frequency-hopping circuit. This makes the development board an excellent tool to attempt to improve the performance of the SINCGARS, perhaps including size and weight reduction.

C. RECOMMENDATIONS

There are many possibilities for follow-on work. MCTSSA would like to improve the RT-1523C operating range when using the standard 3- and 10-foot antennas. This thesis did not examine any other hardware possibilities save the wire connection from the transmit to receive ports.

One option is to change the design parameters of this design to send and receive SINCGARS signals. The FPGA-RT can then be evaluated for bit error rate performance and range. The results can then be compared to the RT-1523C performance and range.

Another possible continuation of this thesis would be to conduct error analysis on the performance of this implementation using the standard RT-1523C antennas. Depending on performance, modifications can be done to the design to determine the most efficient use of the antennas to support the desired results from the RT.

A third option is to conduct bit error analysis on this design and modify it to reduce errors and as a culmination, compare the FPGA performance to the RT-1523C recorded performance. The efficiency and usefulness of this option can further be enhanced by including the use of the 3- and 10-foot RT-1523C antennas.

Lastly, additional work can be done in the actual design of the receiver-transmitter implementation. The SINCGARS can perform frequency-hopping, RS coding and decoding. None of these properties were implemented in the current design and further work can be done to incorporating these functions onto the current design.

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX A. ERRORS ENCOUNTERED AND LESSONS LEARNED

This appendix discusses the errors made and lessons learned during the design process.

A. DESIGN FLOW

Although the design flow displayed in Figure 6 was predominantly used, there were also two other designs flows that were used but found ineffective. One of these design flows was using Simulink for design entry and the DSP Builder for model to VHDL conversion, and then performing all the synthesis, fitting and programming in the Quartus II software; this was not used often because two different programs had to be used and the Quartus II electronic design automation (EDA) tool does not convert the VHDL to schematic files for design modifications. The other possible design flow was the DSP Builder design flow. The main reason the DSP Builder design flow was not used was because the DSP Builder cannot program a device with files that contain intellectual property (IP) in the design. The difference between the DSP Builder design flow and the design flow used is that the design flow used can program using IPs via the use of the Quartus II programmer from the Quartus II software. [17][18]

B. TIMING

The timing errors encountered in this design were mostly due to lack of understanding of the Altera blockset. The SignalCompiler does not allow the use of multiple sample times. If the design contained sample times discrepancies, an error was displayed during compilation. This problem was corrected by using a global sample time, and using the Altera increment/decrement block to create dif-

ferent frequencies that are multiples of the system clock frequency. The counter frequency can be determined using the equation: $f_{counter} = \frac{f_{clock}}{2^N} = \frac{80\ MHz}{2^N}$, where N is the number of bits. [17]

C. SUBSYSTEM

The subsystem feature of Simulink, which permits hierarchical designs, is also used by the DSP Builder. To facilitate this option, the DSP Builder library has an HDL subsystem block that allows the designer to create an Altera subsystem. One of the difficulties is that the HDL subsystem block is linked to the library, so in order to modify the block, it must be unlinked from the library. When the prompt to unlink is displayed, shown in Figure 32, it seems as if an error has been made. But that is not the case. The designer must disable the link in order to modify the block and use it in the design. Once the subsystem has been created using the HDL subsystem block, it must not be re-linked to the library. If the new subsystem is re-linked it changes the HDL subsystem in the library, which is not desired. [17] [24]

Figure 32. "Disable Link" Message.

Another option is to create a subsystem by selecting the part of the model that will be a subsystem, right clicking in the selected area and selecting the "Create subsystem" option. Once the subsystem has been created, the Altera subsystem mask can be added and the SignalCom-

piler will recognize the subsystem. If this option is used, care should be taken to use the Altera busses and ports so the subsystem can be converted to VHDL by the compiler, otherwise the design will not be able to compile. [17] [24]

D. BIT MANIPULATION

The design of the transmitters had one particular issue of concern, the output signal format. In order to transmit the created signal, the DSP board requires that the signal be unsigned. The solution used was to add the value $2^{N-1} - 1$ to the signal, N being the number of bits, in order to convert the signed integer to an unsigned integer. Next, the new signal that was one bit larger than the previous signal was reduced to the 14 bits required by the DAC. This was done by removing the middle bits, thereby keeping the smoothness of the curve at the extremities. [9] [15]

E. SIGNAL ROUTING TAGS

A significant design event was the discovery of the DSP Builder compatibility with the Simulink GOTO and FROM signal routing tags [17]. The GOTO and FROM tags are signal routing blocks that permit signals used in a design to simplify the schematic [24]. They proved to be very useful because they reduced the amount of lines in the design making it less cluttered and easier to examine.

F. IP USAGE

As mentioned in the DESIGN FLOW chapter, the combination of the transmitter and receiver designs caused problems during hardware implementation and required the removal of redundant NCOs. This issue brings up an important point in IP use in designs; if the IP is a source that can be used for multiple purposes, as is the NCO, do not imple-

ment the same IP multiple times in a design. This causes compilation problems, errors, and prevents device programming. Since most IPs are coders, decoders, filters and transforms this issue is not a problem with most IPs, but in case errors occur at compilation, the problem may be the use of redundant IPs. [17]

1. Altera NCO Problems

The first iteration of the BFSK-RT contained a total of four NCOs, two in the transmitter and two in the receiver. The NCOs were actually two pairs of 5 MHz and 10 MHz oscillators, one pair to create the transmitted waveform and the other to correlate the received waveform. This design created hierarchy errors when compiling the design. The cause is the method Quartus II uses to fit designs into hardware. Since two NCOs have the same exact parameters, the fitter attempts to merge them. The error occurs because of the location of the NCOs in the design; they are in different branches of the hierarchy. Since the fitter first merges the NCOs and then attempts to route the signals, the compilation has errors trying to determine where to place the new combined hierarchy and the compilation fails. [17]

G. IMPLEMENTATION ISSUES

Most of the implementation errors and concerns were derived from IP use in the design. Since the NCO IP was used to create and detect the BFSK signal, several problems were encountered when trying to implement the design into hardware because of software compatibility and DSP Builder programming limitations.

1. Software Compatibility

The problem came from the NCO IP and the compatibility of the IP with the Quartus II version that was included with the DSP development kit. When IPs were initially created, they were designed to evaluate design in software only. To use IPs in hardware testing, a license had to be requested from Altera. After the system testing was completed, the license had to be purchased to implement the design in stand alone solutions. Last year, all the IPs and Quartus II were updated to take out a licensing step, and allow designers to perform software and hardware testing with the same license. In order to take advantage of these properties, Quartus II version 4.2 and DSP Builder version 2.2 are required. The DSP development kit was purchased with Quartus II version 3.1 and DSP Builder version 1.2, a combination that is not capable of performing hardware testing. These problems were solved by upgrading both the DSP Builder and the Quartus II software, with some help from the Altera helpdesk. [19]

2. Programming with IPs

Once the IP compatibility issue was solved, the next issue that was addressed was actually programming the FPGA. The SignalCompiler runs from within the Simulink software and has the ability to compile, synthesize, fit and program the design onto the DSP board. This capability is limited to non-IP designs though. In order to program designs that contain IPs, it is necessary to program the board using the Quartus II programmer so the FPGA can be operated in the OpenCore tethered mode. Once this limitation was determined, the solution was to compile the design usinjg the SignalCompiler and then program the FPGA using the Quartus II programmer. [19]

This Appendix discussed the errors encountered and lessons learned during the design flow. Appendix B displays the models that comprise the FPGA BFSK-RT.

APPENDIX B. BFSK-RT SIMULINK MODELS

This appendix contains all the Simulink models for the BFSK-RT. Although all the majority of these models are in the body of the thesis, this appendix provides *all* the models that were used in the design in one place for easy reference.

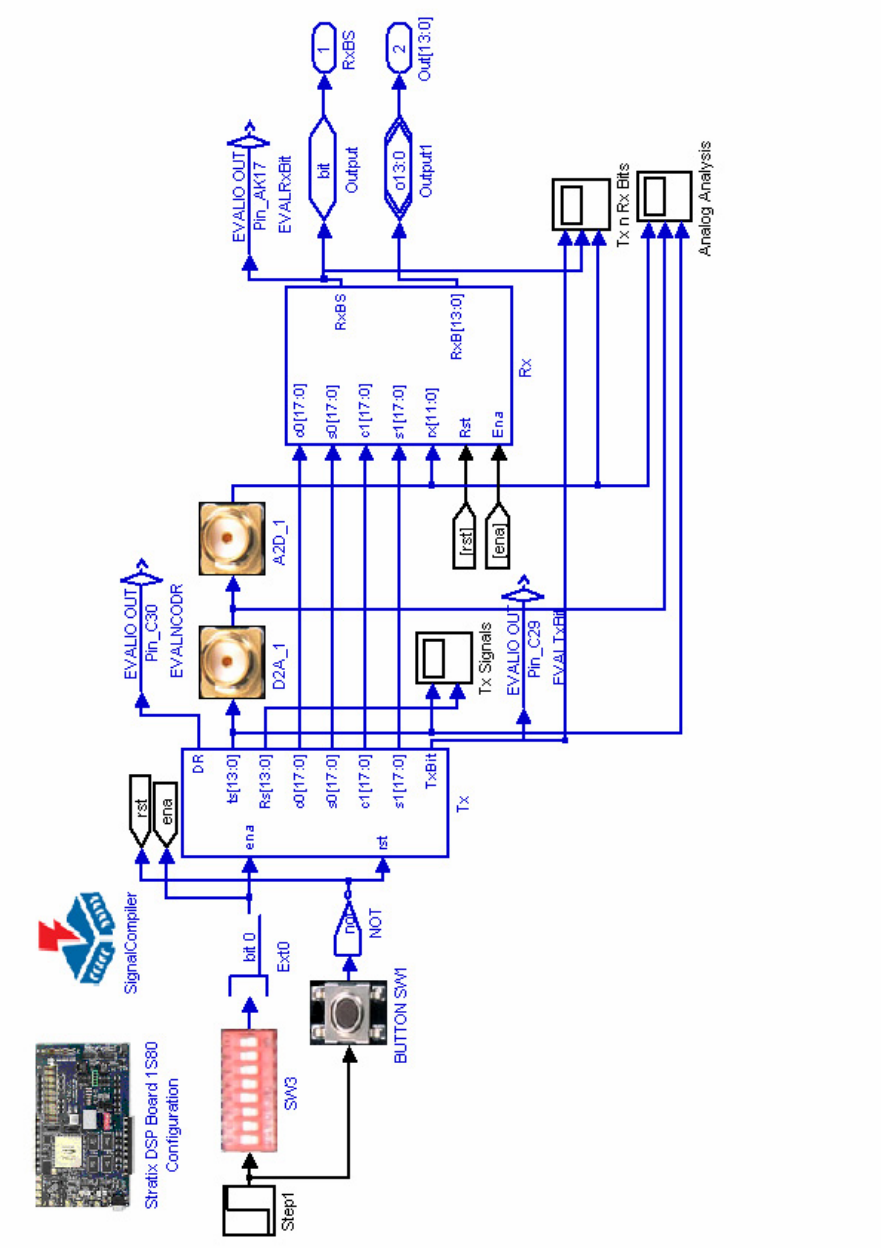

Figure 33. BFSK-RT Top-level Model.

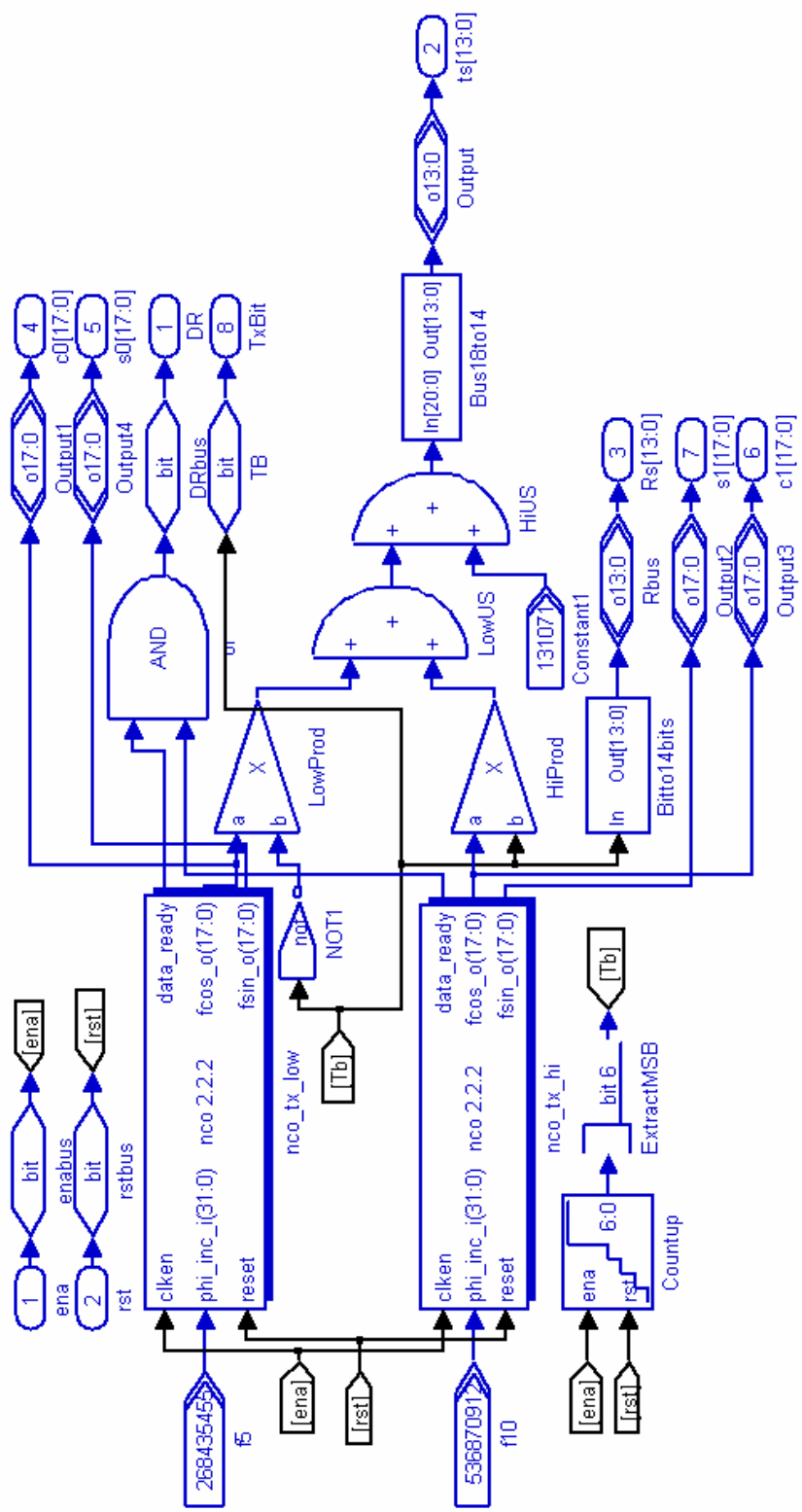

Figure 34. BFSK Transmitter Model.

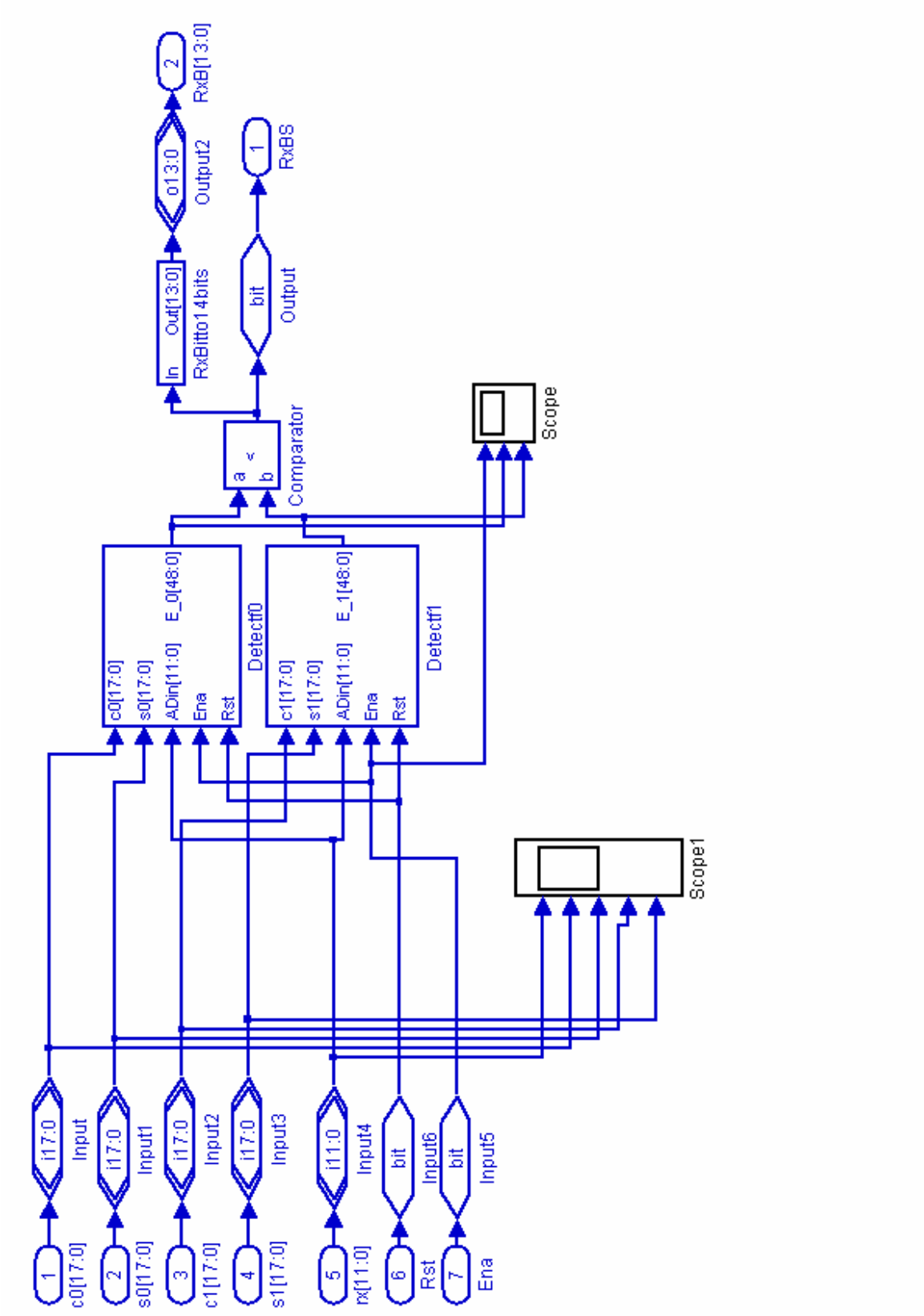

Figure 35. BFSK Receiver Model.

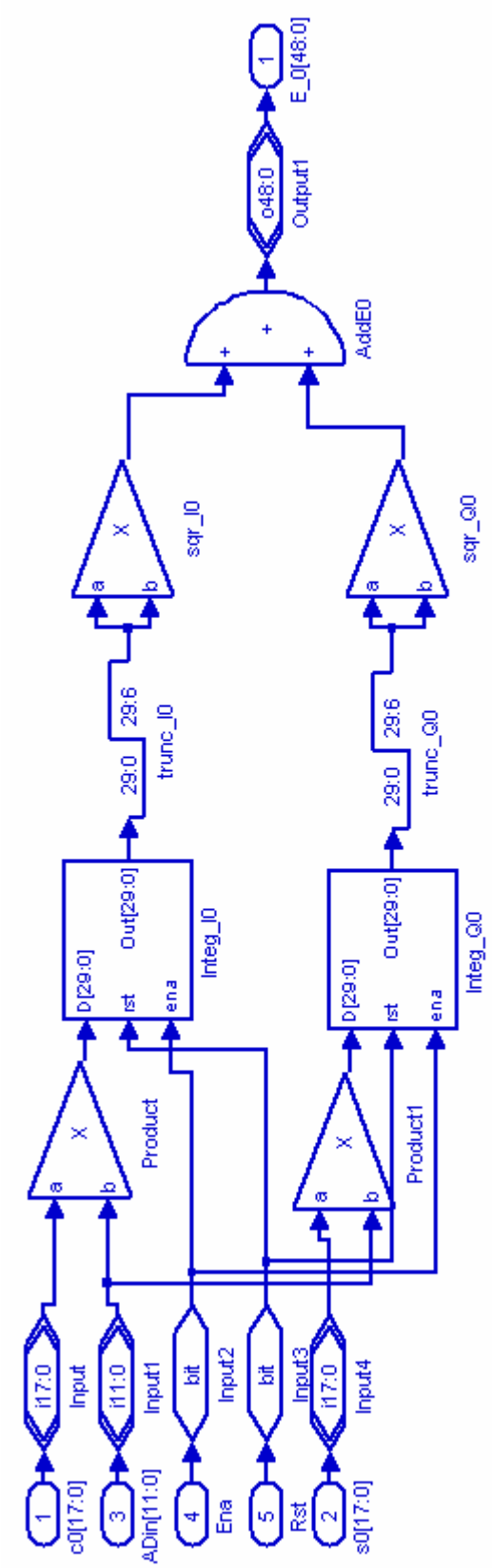

Figure 36. Detect-0 Model.

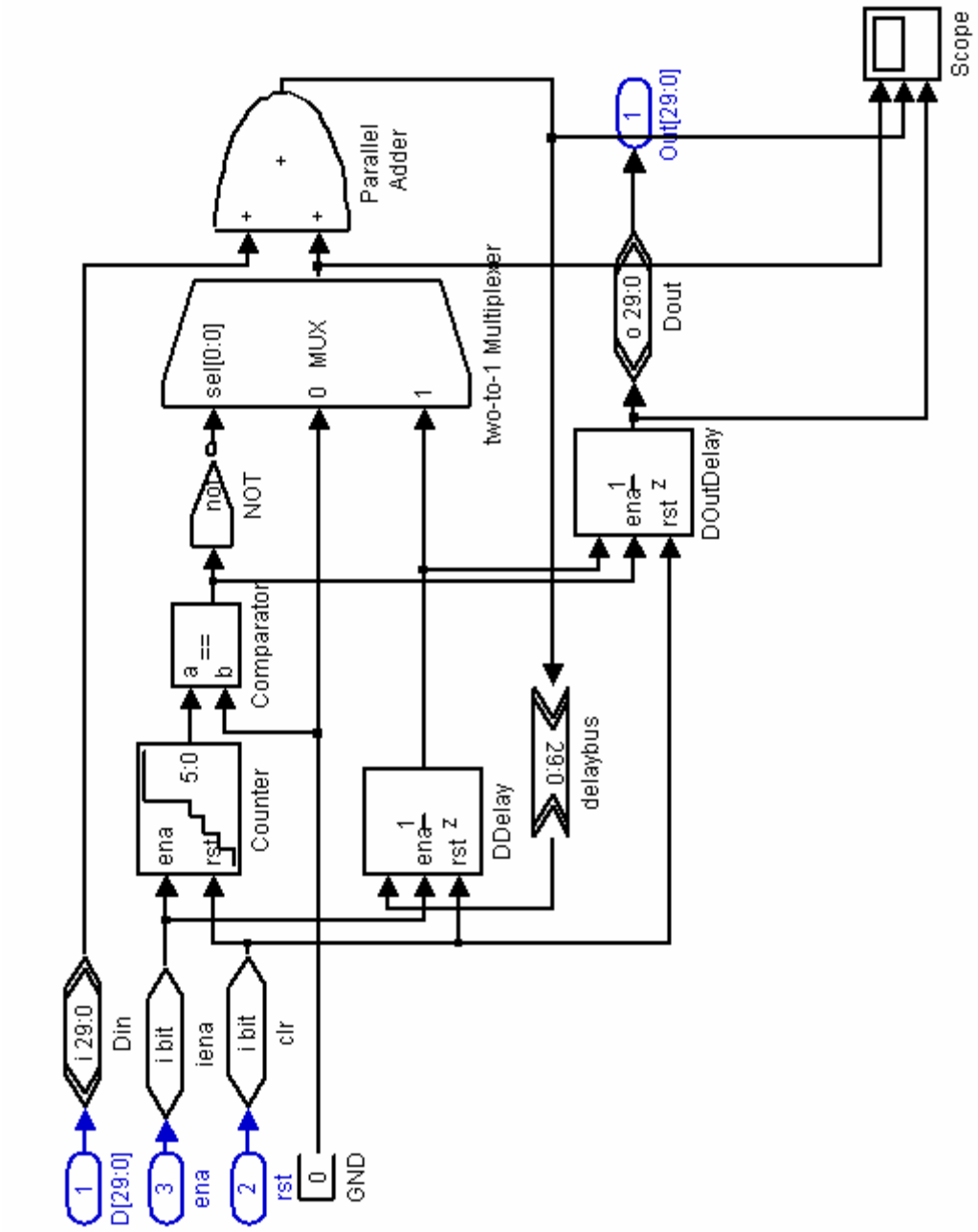

Figure 37. Integrator Model.

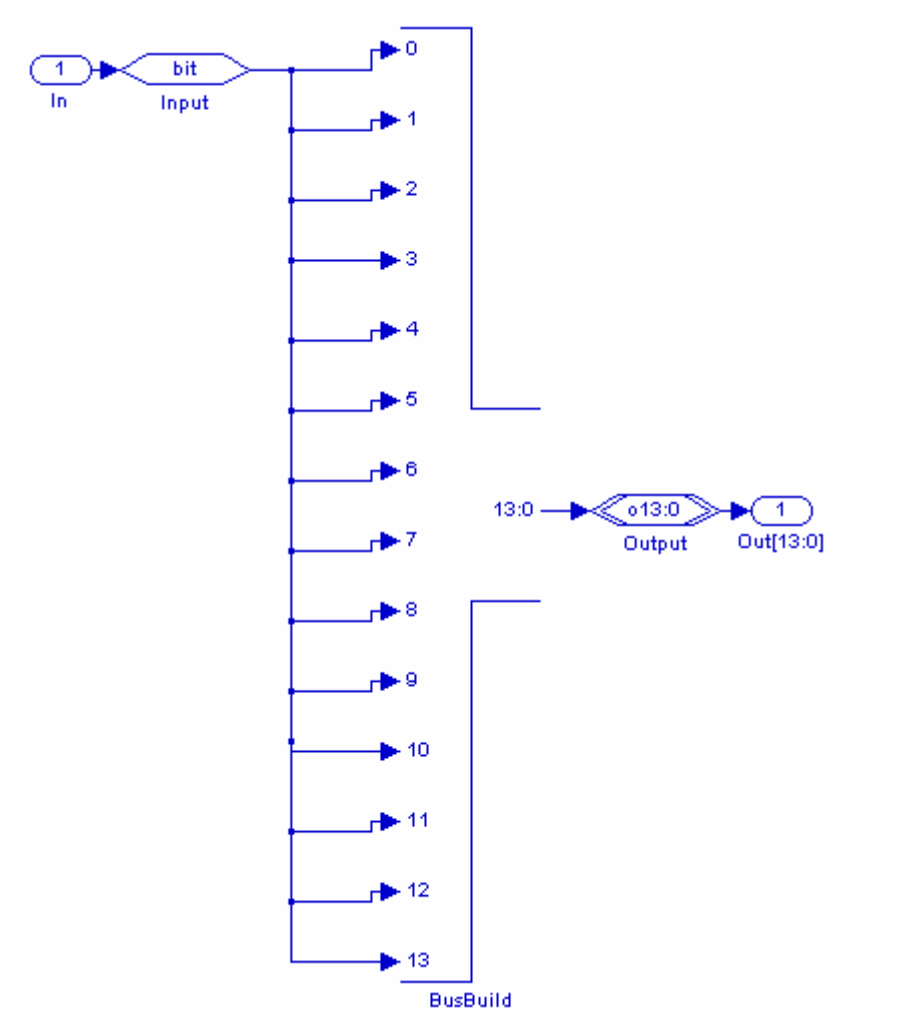

Figure 38. Bit to 14-bit Buss Model.

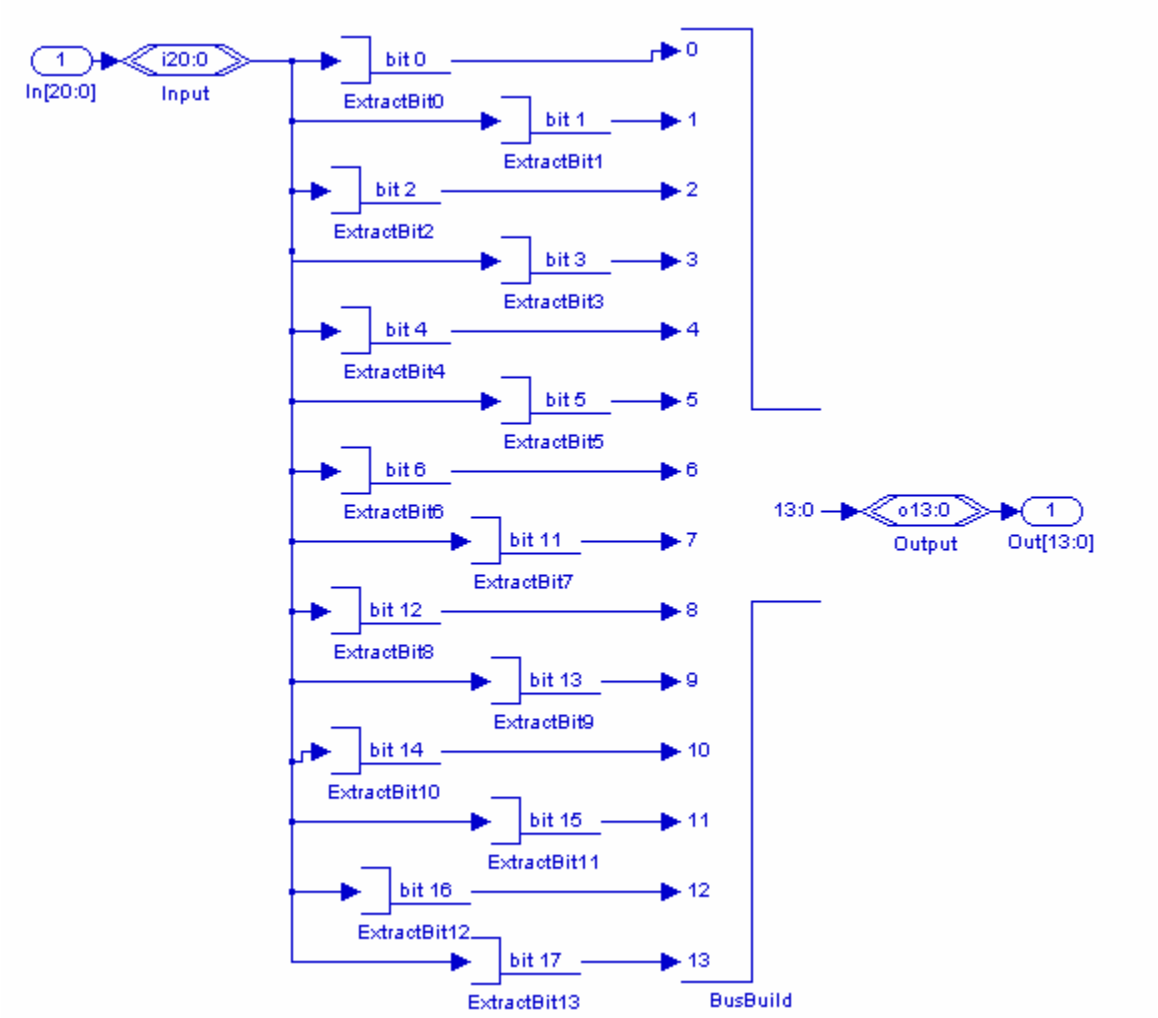

Figure 39. 18-bit to 14-bit Buss Model.

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF REFERENCES

1. U.S. Department of Defense, U.S. Marine Corps, *MCRP 6-2.2.2 Talk II – SINCGARS: Multiservice Communications Procedures for the Single-Channel Ground and Airborne Radio System*, pp. ii - I-14, GPO, Washington, DC, 1996.

2. Dan Caterinicchia with Matthew French, “Network-centric warfare: Not there yet,” *Federal Computer Week*, June 9, 2003.

3. “DSP Development Kit, Stratix Edition,” [http://www.altera.com/products/devkits/altera/kit-dsp_stratix.html], last accessed on March 05, 2005.

4. “SINCGARS: Evolution to Revolution,” ITT Industries, pp. 1-6, Ft Wayne, IN, 1997.

5. Bradley J. Hamilton, “SINCGARS System Improvement Program (SIP) specific radio improvements,” *Tactical Communications Conference,* pp. 397-406, 1996.

6. Max Green, “SINCGARS Signal Output Power Test: Test Report – DRAFT,” p. 3, MCTSSA, Camp Pendelton, CA, August 18, 2004.

7. “Joint Tactical Radio System – JTRS,” [http://jtrs.army.mil/index.htm], last accessed on March 05, 2005.

8. Richard Paradise, “Modeling and Simulation of the Physical Layer of the Single Channel Ground and Airborne Radio System (SINCGARS),” Master’s thesis, Naval Postgraduate School, 2005.

9. John F. Wakerly, *Digital Design: Principles and Practices*, Prentice Hall, NJ, 2001.

10. Michael Barr, "Programmable Logic: What's it to Ya?" *Embedded Systems Programming*, pp. 75-84, June 1999.

11. “Xilinx Home,” [http://www.xilinx.com/xlnx/xil_prodcat_landingpage.jsp?sSecondary-NavPick=null&sGlobalNavPick=PRODUCTS&iLanguageID=1], last accessed on April 14, 2005.

12. “Altera Home,” [http://www.altera.com/], last accessed on May 31, 2005.

13. “Stratix Devices,” [http://www.altera.com/products/devices/stratix/stx-index.jsp], last accessed on May 31, 2005.

14. “Stratix Device Handbook Vol. 1-2,” Technical Reports S5V1-3.2 – S5V2-3.3, Altera Corp., San Jose, CA, September 2004.

15. “Stratix EP1S80 DSP Development Board Data Sheet,” Technical Report DS-STXDVBD-1.2, Altera Corp., San Jose, CA, July 2003.

16. “Simulink 6.2,” [http://www.mathworks.com/products/simulink/], last accessed on March 11, 2005.

17. “DSP Builder User Guide,” Technical Report UG-DSPBUILDER-3.0, Altera Corp., pp. 1–1 — 5-22, San Jose, CA, August 2004.

18. “Quartus II Handbook Vol. 1-4,” Technical Reports qii5v1-3.1 – qii5v4-1.0, Altera Corp., San Jose, CA, December 2004.

19. “Application Note 320: OpenCore Plus Evaluation of Megafunctions,” Technical Report AN-320-1.2, Altera Corp., pp. 1-5, San Jose, CA, June 2004.

20. U.S. Department of Defense, U.S. Marine Corps. *TM 5820-45&P/1-1 Volume I, U.S. Marine Corps Technical Manual, Intermediate and Depot Maintenance, Single Channel Ground and Airborne Radio System (SINCGARS)*, pp. 1-3 – 3-131, GPO, Washington, DC, 1997.

21. Bernard Sklar, *Digital Communications: Fundamentals and Applications,* 2d ed., pp. 2-250, Prentice Hall, NJ, 2001.

22. Martin S. Roden, *Analog and Digital Communication Systems,* 5th ed., pp. 62-369, Discovery Press, Los Angeles, CA, 2003.

23. “NCO Compiler: MegaCore Function User Guide,” Technical Report UG-NCOCOMPILER-2.3, Altera Corp., pp. 1-1 – A-6, San Jose, CA, June 2004.

24. “Using Simulink, Version 6,” Technical Report Using Simulink, The Mathworks Inc., pp. 1-1 — 12-34, Natick, MA, 2005.

THIS PAGE INTENTIONALLY LEFT BLANK

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center
   Ft. Belvoir, Virginia

2. Dudley Knox Library
   Naval Postgraduate School
   Monterey, California

3. Marine Corps Representative
   Naval Postgraduate School
   Monterey, California

4. Director, Training and Education, MCCDC, Code C46
   Quantico, Virginia

5. Director, Marine Corps Research Center, MCCDC, Code C40RC
   Quantico, Virginia

6. Marine Corps Tactical Systems Support Activity (Attn: Operations Officer)
   Camp Pendleton, California

7. Frank Kragh
   Naval Postgraduate School
   Monterey, California

8. Herschel Loomis
   Naval Postgraduate School
   Monterey, California

9. Clark Robertson
   Naval Postgraduate School
   Monterey, California

10. Nathan Beltz
    Naval Postgraduate School
    Monterey, California

11. Capt Max Green
    Marine Corps Tactical Systems Support Activity
    Camp Pendleton, California

12. Capt Juan Svenningsen
Marine Corps Systems Command
Quantico, Virginia

13. Capt Richard Paradise
Marine Corps Systems Command
Quantico, Virginia